经典计算机科学著作最新版

计算机程序设计艺术

第4卷 第4册（双语版）

生成所有树
组合生成的历史

The Art of Computer Programming, Volume 4
Generating All Trees
History of Combinatorial Generation

Fascicle 4

苏运霖 译

（美）Donald E. Knuth 著

机械工业出版社
China Machine Press

关于算法分析的这套多卷论著已经长期被公认为经典计算机科学的定义性描述。《计算机程序设计艺术，第4卷 组合算法》是作者近期发表的部分内容。作为关于组合查找的冗长一章的一部分，这个分册讨论生成所有树和组合生成的历史。读者从本册中不仅会看到很多新内容，而且会发现与第1卷～第3卷及计算机科学和数学领域的丰富联系。一如既往，书中包括了大量的习题和富有挑战性的难题。

Simplified Chinese edition copyright © 2007 by Pearson Education Asia Limited and China Machine Press.

Original English language title: *The Art of Computer Programming, Volume 4, Fascicle 4, Generating All Trees; History of Combinatorial Generation* (0-321-33570-8) by Donald E. Knuth , Copyright© 2006.

All rights reserved.

Published by arrangement with the original publisher, Pearson Education, Inc., publishing as ADDISON WESLEY.

本书封面贴有Pearson Education（培生教育出版集团）激光防伪标签，无标签者不得销售。

版权所有，侵权必究。

本书法律顾问　北京市展达律师事务所

本书版权登记号：图字：01-2006-1931

图书在版编目（CIP）数据

计算机程序设计艺术 第4卷 第4册 生成所有树-组合生成的历史（双语版）/（美）克努特（Knuth, D. E.）编著；苏运霖译．－北京：机械工业出版社，2007.4

书名原文：The Art of Computer Programming, Volume 4, Fascicle 4：Generating All Trees, History of Combinatorial Generation

ISBN 978-7-111-20825-9

Ⅰ.计…　Ⅱ.①克…　②苏…　Ⅲ.程序设计－英、汉　Ⅳ.TP311.1

中国版本图书馆CIP数据核字（2007）第013348号

机械工业出版社（北京市西城区百万庄大街22号　邮政编码　100037）

责任编辑：王　玉

北京瑞德印刷有限公司印刷·新华书店北京发行所发行

2007年4月第1版第1次印刷

170mm×242mm · 15.75印张

定价：42.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线：（010）68326294

PREFACE

I like to work in a variety of fields
in order to spread my mistakes more thinly.
— VICTOR KLEE (1999)

THIS BOOKLET is Fascicle 4 of *The Art of Computer Programming*, Volume 4: *Combinatorial Algorithms*. As explained in the preface to Fascicle 1 of Volume 1, I'm circulating the material in this preliminary form because I know that the task of completing Volume 4 will take many years; I can't wait for people to begin reading what I've written so far and to provide valuable feedback.

To put the material in context, this fascicle contains Sections 7.2.1.6 and 7.2.1.7 of a long, long chapter on combinatorial searching. Chapter 7 will eventually fill three volumes (namely Volumes 4A, 4B, and 4C), assuming that I'm able to remain healthy. It will begin with a short review of graph theory, with emphasis on some highlights of significant graphs in the Stanford GraphBase, from which I will be drawing many examples. Then comes Section 7.1, which deals with bitwise manipulation and with algorithms relating to Boolean functions. Section 7.2 is about generating all possibilities, and it begins with Section 7.2.1: Generating Basic Combinatorial Patterns. Details about various useful ways to generate n-tuples, permutations, combinations, and partitions appear in Sections 7.2.1.1 through 7.2.1.5. That sets the stage for the main contents of the present booklet, namely Section 7.2.1.6, which completes the study of basic patterns by discussing how to generate various kinds of tree structures; and Section 7.2.1.7, which completes the story of the preceding subsections by discussing the origins of the concepts and pointing to other sources of information. Section 7.2.2 will deal with backtracking in general. And so it will go on, if all goes well; an outline of the entire Chapter 7 as currently envisaged appears on the `taocp` webpage that is cited on page ii.

I had great pleasure writing this material, akin to the thrill of excitement that I felt when writing Volume 2 many years ago. As in Volume 2, where I found to my delight that the basic principles of elementary probability theory and number theory arose naturally in the study of algorithms for random number generation and arithmetic, I learned while preparing Section 7.2.1 that the basic principles of elementary combinatorics arise naturally and in a highly motivated way when we study algorithms for combinatorial generation. Thus, I found once again that a beautiful story was "out there" waiting to be told.

In fact, I've been looking forward to writing about the generation of trees for a long time, because tree structures have a special place in the hearts of

all computer scientists. Although I certainly enjoyed preparing the material about classic combinatorial structures like tuples, permutations, combinations, and partitions in Sections 7.2.1.1–7.2.1.5, the truth is that I've saved the best for last: Now it's time for the dessert course. Ever since 1994 I've been giving an annual "Christmas tree lecture" at Stanford University, to talk about the most noteworthy facts about trees that I learned during the current year, and at last I am able to put the contents of those lectures into written form. This topic, like many desserts, is extremely rich, yet immensely satisfying. The theory of trees also ties together a lot of concepts from different aspects of computer programming.

And Section 7.2.1.7, about the history of combinatorial generation, was equally satisfying to the other half of my brain, because it involves poetry, music, religion, philosophy, logic, and intellectual pastimes from many different cultures in many different parts of the world. The roots of combinatorial thinking go very deep, and I can't help but think that I learned a lot about human beings in general as I was putting the pieces of this story together.

My original intention was to devote far less space to such subjects. But when I saw how fundamental the ideas were, I knew that I could never be happy unless I covered the basics quite thoroughly. Therefore I've done my best to build a solid foundation of theoretical and practical ideas that will support many kinds of reliable superstructures.

I thank Frank Ruskey for bravely foisting an early draft of this material on college students and for telling me about his classroom experiences. Many other readers have also helped me to check the first drafts, especially in Section 7.2.1.7 where I was often operating at or beyond the limits of my ability to understand languages other than English.

I shall happily pay a finder's fee of \$2.56 for each error in this fascicle when it is first reported to me, whether that error be typographical, technical, or historical. The same reward holds for items that I forgot to put in the index. And valuable suggestions for improvements to the text are worth 32¢ each. (Furthermore, if you find a better solution to an exercise, I'll actually reward you with immortal glory instead of mere money, by publishing your name in the eventual book:–)

Cross references to yet-unwritten material sometimes appear as '00' in the following pages; this impossible value is a placeholder for the actual numbers to be supplied later.

Happy reading!

Stanford, California D. E. K.
June 2005

A note on notation. At the beginning of Chapter 7 I'll define some operations on graphs for which many different notations are presently rampant. My current plan is to say that, if G is a graph on the vertices $U = \{u_1, \ldots, u_m\}$ and if H is a graph on the vertices $V = \{v_1, \ldots, v_n\}$, then:

- $G + H$ is the sum, aka juxtaposition, of G and H: It has the $m + n$ vertices $U \cup V$ and the edges of G and H.
- $G \mathbin{\overline{+}} H$ is the cosum, aka join, of G and H, namely the complement of the juxtaposition of their complements. (Thus its edges are those of G and H, plus all $u_j \,—\, v_k$.)
- $G \times H$ is the Cartesian product of G and H: It has the mn vertices $U \times V$; its edges are $(u,v) \,—\, (u',v)$ when $u \,—\, u'$ in G, and $(u,v) \,—\, (u,v')$ when $v \,—\, v'$ in H.
- $G \diamond H$ is the direct product, aka conjunction, of G and H: Again its vertices are $U \times V$, but its edges are $(u,v) \,—\, (u',v')$ if and only if $u \,—\, u'$ in G and $v \,—\, v'$ in H.
- $G \boxtimes H$ is the strong product of G and H: As its symbol implies, it combines the edges of $G \times H$ and $G \diamond H$.
- There also are coproducts, analogous to the cosum.

Other notations that are used in this fascicle and not otherwise explained can be found in the Index to Notations at the end of Volumes 1, 2, or 3. Those indexes point to the places where further information is available. Of course Volume 4 will some day contain its own Index to Notations.

CONTENTS

Preface ······ *III*

Chapter 7 Combinatorial Searching

7.2 **Generating All Possibilities** ······ *120*
  7.2.1 Generating Basic Combinatorial Patterns ······ *120*
    7.2.1.1 Generating all *n*-tuples ······ *120*
    7.2.1.2 Generating all permutations ······ *120*
    7.2.1.3 Generating all combinations ······ *120*
    7.2.1.4 Generating all partitions ······ *120*
    7.2.1.5 Generating all set partitions ······ *120*
    7.2.1.6 Generating all trees ······ *121*
    7.2.1.7 History and further references ······ *162*

Answers to Exercises ······ *188*
Index and Glossary ······ *227*

目 录

译者序 ······ *115*
前言 ······ *117*

第7章 组合查找

7.2 **生成所有可能性** ······ *120*
  7.2.1 生成基本的组合模式 ······ *120*
    7.2.1.1 生成所有*n*元组 ······ *120*
    7.2.1.2 生成所有排列 ······ *120*
    7.2.1.3 生成所有组合 ······ *120*
    7.2.1.4 生成所有分划 ······ *120*
    7.2.1.5 生成所有集合的分划 ······ *120*
    7.2.1.6 生成所有树 ······ *121*
    7.2.1.7 历史和进一步的参考文献 ······ *162*

习题答案 ······ *188*
索引和词汇表 ······ *227*

CHAPTER SEVEN

COMBINATORIAL SEARCHING

The opening sections of this chapter will appear in Volume 4, Fascicle 0, and Volume 4, Fascicle 1, planned for publication in 2006 and 2007.

7.2. GENERATING ALL POSSIBILITIES

7.2.1. Generating Basic Combinatorial Patterns

OUR GOAL in this section is to study methods for running through all of the possibilities in some combinatorial universe, because we often face problems in which an exhaustive examination of all cases is necessary or desirable. ...

7.2.1.1. Generating all n-tuples. Let's start small, by considering how to run through all 2^n strings that consist of n binary digits. ...

7.2.1.2. Generating all permutations. After n-tuples, the next most important item on nearly everybody's wish list for combinatorial generation is the task of visiting all *permutations* of some given set or multiset. ...

The complete texts of Sections 7.2.1.1 and 7.2.1.2 can be found in Volume 4, Fascicle 2, first published in February 2005.

7.2.1.3. Generating all combinations. Combinatorial mathematics is often described as "the study of permutations, combinations, etc.," so we turn our attention now to combinations. ...

7.2.1.4. Generating all partitions. Richard Stanley's magnificent book *Enumerative Combinatorics* (1986) begins by discussing The Twelvefold Way, a $2 \times 2 \times 3$ array of basic combinatorial problems that arise frequently in practice. ... We've learned about n-tuples, permutations, combinations, and compositions in previous sections of this chapter; so we can complete our study of classical combinatorial mathematics by learning about the remaining entries in Stanley's table, which all involve *partitions*. ...

The *partitions of an integer* are the ways to write it as a sum of positive integers, disregarding order. ...

7.2.1.5. Generating all set partitions. Now let's shift gears and concentrate on a rather different kind of partition. The *partitions of a set* are the ways to regard that set as a union of nonempty, disjoint subsets. ...

The complete texts of Sections 7.2.1.3, 7.2.1.4, and 7.2.1.5 can be found in Volume 4, Fascicle 3, first published in July 2005.

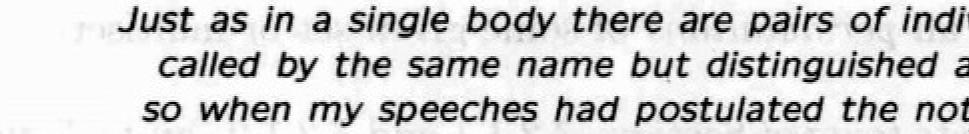
Explain the significance of the following sequence:
un, dos, tres, quatre, cinc, sis, set, vuit, nou, deu, ...
— RICHARD P. STANLEY, *Enumerative Combinatorics* (1999)

Just as in a single body there are pairs of individual members,
called by the same name but distinguished as right and left,
so when my speeches had postulated the notion of madness,
as a single generic aspect of human nature,
the speech that divided the left-hand portion
repeatedly broke it down into smaller and smaller parts.
— SOCRATES, *Phædrus* 266A (c. 370 B.C.)

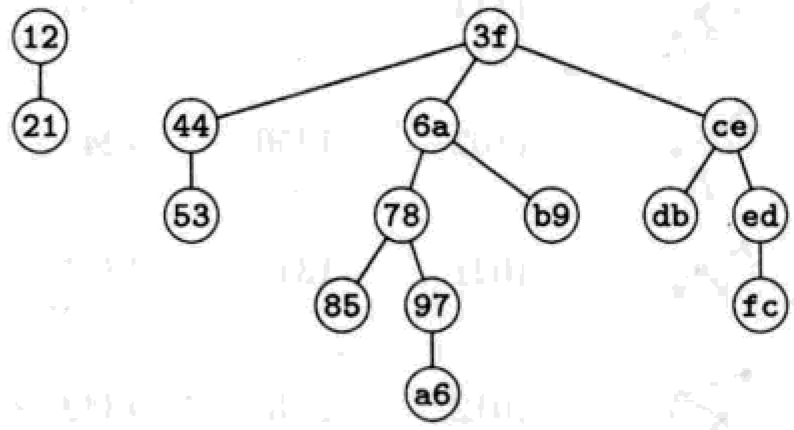
7.2.1.6. Generating all trees. We've now completed our study of the classical concepts of combinatorics: tuples, permutations, combinations, and partitions. But computer scientists have added another fundamental class of patterns to the traditional repertoire, namely the hierarchical arrangements known as trees. Trees sprout up just about everywhere in computer science, as we've seen in Section 2.3 and in nearly every subsequent section of *The Art of Computer Programming*. Therefore we turn now to the study of simple algorithms by which trees of various species can be exhaustively explored.

First let's review the basic connection between nested parentheses and forests of trees. For example,

```
1 2   3 4 5   6 7 8   9 a     b   c d   e f
( ( ) ) ( ( ( ) ) ( ( ( ) ( ( ) ) ) ( ) ) ( ( ) ( ( ) ) ) )      (1)
    1 2       3 4       5     6 7 8   9 a     b     c d e f
```

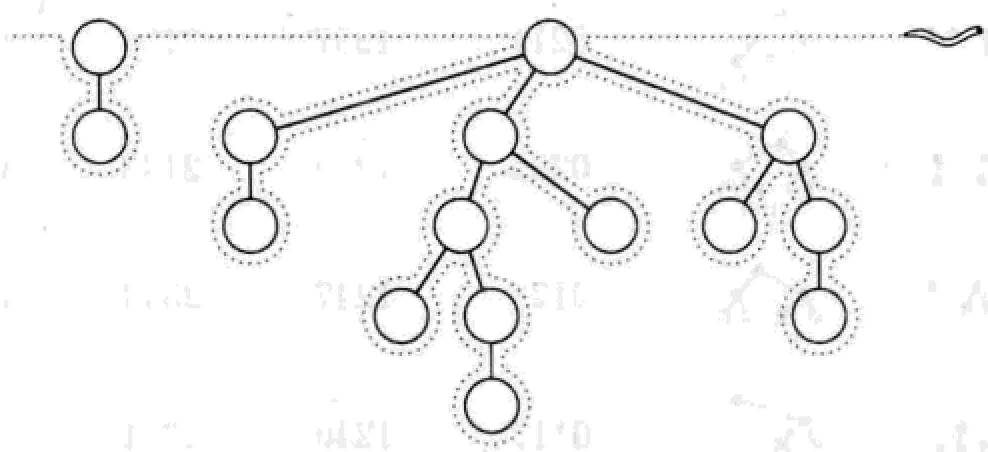


illustrates a string containing fifteen left parens '(' labeled `1`, `2`, ..., `f`, and fifteen right parens ')' also labeled `1` through `f`; gray lines beneath the string show how the parentheses match up to form fifteen pairs `12`, `21`, `3f`, `44`, `53`, `6a`, `78`, `85`, `97`, `a6`, `b9`, `ce`, `db`, `ed`, and `fc`. This string corresponds to the forest



(2)

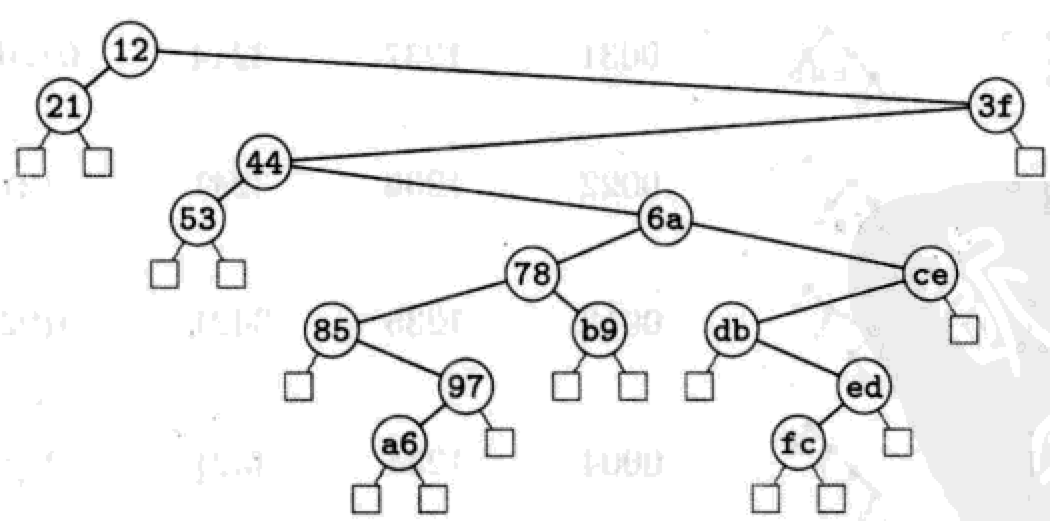
in which the nodes are (12), (21), (3f), ..., (fc) in preorder (sorted by first coordinates) and (21), (12), (53), ..., (3f) in postorder (sorted by second coordinates). If we imagine a worm that crawls around the periphery of the forest,



(3)

seeing a '(' whenever it passes the left edge of a node and a ')' whenever it passes a node's right edge, that worm will have reconstructed the original string (1).

The forest in (2) corresponds, in turn, to the binary tree

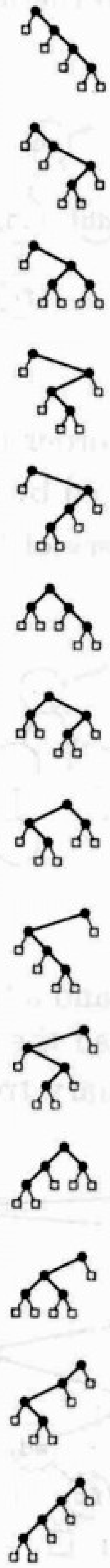


(4)

via the "natural correspondence" discussed in Section 2.3.2; here the nodes are (21), (12), (53), ..., (3f) in *symmetric* order, also known as inorder. The left subtree of node (x) in the binary tree is the leftmost child of (x) in the forest, or it is an "external node" □ if (x) is childless. The right subtree of (x) in the binary tree is its right sibling in the forest, or □ if (x) is the rightmost child in its family. Roots of the trees in the forest are considered to be siblings, and the leftmost root of the forest is the root of the binary tree.

Table 1

NESTED PARENTHESES AND RELATED OBJECTS WHEN $n = 4$


$a_1a_2\ldots a_8$	forest	binary tree	$d_1d_2d_3d_4$	$z_1z_2z_3z_4$	$p_1p_2p_3p_4$	$c_1c_2c_3c_4$	matching
()()()()			1111	1357	1234	0000	
()()(())			1102	1356	1243	0001	
()(())()			1021	1347	1324	0010	
()(()())			1012	1346	1342	0011	
()((()))			1003	1345	1432	0012	
(())()()			0211	1257	2134	0100	
(())(())			0202	1256	2143	0101	
(()())()			0121	1247	2314	0110	
(()()())			0112	1246	2341	0111	
(()(()))			0103	1245	2431	0112	
((()))()			0031	1237	3214	0120	
((())())			0022	1236	3241	0121	
((()()))			0013	1235	3421	0122	
(((())))			0004	1234	4321	0123	

A string $a_1a_2\ldots a_{2n}$ of parentheses is properly nested if and only if it contains n occurrences of '(' and n occurrences of ')', where the kth '(' precedes the kth ')' for $1 \le k \le n$. The easiest way to explore all strings of nested parentheses is to visit them in lexicographic order. The following algorithm, which considers ')' to be lexicographically smaller than '(', includes some refinements for efficiency suggested by I. Semba [*Inf. Processing Letters* **12** (1981), 188–192]:

Algorithm P (*Nested parentheses in lexicographic order*). Given an integer $n \geq 2$, this algorithm generates all strings $a_1a_2 \ldots a_{2n}$ of nested parentheses.

P1. [Initialize.] Set $a_{2k-1} \leftarrow$ '(' and $a_{2k} \leftarrow$ ')' for $1 \leq k \leq n$; also set $a_0 \leftarrow$ ')' and $m \leftarrow 2n-1$.

P2. [Visit.] Visit the nested string $a_1a_2 \ldots a_{2n}$. (At this point $a_m =$ '(', and $a_k =$ ')' for $m < k \leq 2n$.)

P3. [Easy case?] Set $a_m \leftarrow$ ')'. Then if $a_{m-1} =$ ')', set $a_{m-1} \leftarrow$ '(', $m \leftarrow m-1$, and return to P2.

P4. [Find j.] Set $j \leftarrow m-1$ and $k \leftarrow 2n-1$. While $a_j =$ '(', set $a_j \leftarrow$ ')', $a_k \leftarrow$ '(', $j \leftarrow j-1$, and $k \leftarrow k-2$.

P5. [Increase a_j.] Terminate the algorithm if $j = 0$. Otherwise set $a_j \leftarrow$ '(', $m \leftarrow 2n-1$, and go back to P2. ▌

We will see later that the loop in step P4 is almost always short: The operation $a_j \leftarrow$ ')' is performed only about $\frac{1}{3}$ times per nested string visited, on the average.

Why does Algorithm P work? Let A_{pq} be the sequence of all strings α that contain p left parentheses and $q \geq p$ right parentheses, where $(^{q-p}\alpha$ is properly nested, listed in lexicographic order. Then Algorithm P is supposed to generate A_{nn}, where it is easy to see that A_{pq} obeys the recursive rules

$$A_{pq} =)\,A_{p(q-1)},\ (\,A_{(p-1)q}, \quad \text{if } 0 \leq p \leq q \neq 0; \qquad A_{00} = \epsilon; \tag{5}$$

also A_{pq} is empty if $p < 0$ or $p > q$. The first element of A_{pq} is $)^{q-p}()\ldots()$, where there are p pairs '()'; the last element is $(^p)^q$. Thus the lexicographic generation process consists of scanning from the right until finding a trailing string of the form $a_j \ldots a_{2n} =)(^{p+1})^q$ and replacing it by $()^{q+1-p}()\ldots()$. Steps P4 and P5 do this efficiently, while step P3 handles the simple case $p = 0$.

Table 1 illustrates the output of Algorithm P when $n = 4$, together with the corresponding forest and binary tree as in (2) and (4). Several other equivalent combinatorial objects also appear in Table 1: For example, a string of nested parentheses can be run-length encoded as

$$()^{d_1}()^{d_2}\ldots()^{d_n}, \tag{6}$$

where the nonnegative integers $d_1d_2\ldots d_n$ are characterized by the constraints

$$d_1 + d_2 + \cdots + d_k \leq k \quad \text{for } 1 \leq k < n; \qquad d_1 + d_2 + \cdots + d_n = n. \tag{7}$$

We can also represent nested parentheses by the sequence $z_1z_2\ldots z_n$, which specifies the indices where the left parentheses appear. In essence, $z_1z_2\ldots z_n$ is one of the $\binom{2n}{n}$ combinations of n things from the set $\{1, 2, \ldots, 2n\}$, subject to the special constraints

$$z_{k-1} < z_k < 2k \qquad \text{for } 1 \leq k \leq n, \tag{8}$$

if we assume that $z_0 = 0$. The z's are of course related to the d's:

$$d_k = z_{k+1} - z_k - 1 \qquad \text{for } 1 \leq k < n. \tag{9}$$

Algorithm P becomes particularly simple when it is rewritten to generate the combinations $z_1z_2\ldots z_n$ instead of the strings $a_1a_2\ldots a_{2n}$. (See exercise 2.)

A parenthesis string can also be represented by the permutation $p_1p_2\ldots p_n$, where the kth right parenthesis matches the p_kth left parenthesis; in other words, the kth node of the associated forest in postorder is the p_kth node in preorder. (By exercise 2.3.2–20, node j is a descendant of node k in the forest if and only if $j<k$ and $p_j>p_k$, when we label the nodes in postorder.) The inversion table $c_1c_2\ldots c_n$ characterizes this permutation by the rule that exactly c_k elements to the right of k are less than k (see exercise 5.1.1–7); allowable inversion tables have $c_1=0$ and

$$0 \le c_{k+1} \le c_k+1 \quad \text{for } 1\le k<n. \tag{10}$$

Moreover, exercise 3 proves that c_k is the level of the forest's kth node in preorder (the depth of the kth left parenthesis), a fact that is equivalent to the formula

$$c_k = 2k-1-z_k. \tag{11}$$

Table 1 and exercise 6 also illustrate a special kind of *matching*, by which $2n$ people at a circular table can simultaneously shake hands without interference.

Thus Algorithm P can be useful indeed. But if our goal is to generate all binary trees, represented by left links $l_1l_2\ldots l_n$ and right links $r_1r_2\ldots r_n$, the lexicographic sequence in Table 1 is rather awkward; the data we need to get from one tree to its successor is not readily available. Fortunately, an ingenious alternative scheme for direct generation of all linked binary trees is also available:

Algorithm B (*Binary trees*). Given $n\ge 1$, this algorithm generates all binary trees with n internal nodes, representing them via left links $l_1l_2\ldots l_n$ and right links $r_1r_2\ldots r_n$, with nodes labeled in preorder. (Thus, for example, node 1 is always the root, and l_k is either $k+1$ or 0; if $l_1=0$ and $n>1$ then $r_1=2$.)

B1. [Initialize.] Set $l_k\leftarrow k+1$ and $r_k\leftarrow 0$ for $1\le k<n$; also set $l_n\leftarrow r_n\leftarrow 0$, and set $l_{n+1}\leftarrow 1$ (for convenience in step B3).

B2. [Visit.] Visit the binary tree represented by $l_1l_2\ldots l_n$ and $r_1r_2\ldots r_n$.

B3. [Find j.] Set $j\leftarrow 1$. While $l_j=0$, set $r_j\leftarrow 0$, $l_j\leftarrow j+1$, and $j\leftarrow j+1$. Then terminate the algorithm if $j>n$.

B4. [Find k and y.] Set $y\leftarrow l_j$ and $k\leftarrow 0$. While $r_y>0$, set $k\leftarrow y$ and $y\leftarrow r_y$.

B5. [Promote y.] If $k>0$, set $r_k\leftarrow 0$; otherwise set $l_j\leftarrow 0$. Then set $r_y\leftarrow r_j$, $r_j\leftarrow y$, and return to B2. ▌

[See W. Skarbek, *Theoretical Computer Science* **57** (1988), 153–159; step B3 uses an idea of J. Korsh.] Exercise 44 proves that the loops in steps B3 and B4 both tend to be very short. Indeed, fewer than 9 memory references are needed, on the average, to transform a linked binary tree into its successor.

Table 2 shows the fourteen binary trees that are generated when $n=4$, together with their corresponding forests and with two related sequences: Arrays $e_1e_2\ldots e_n$ and $s_1s_2\ldots s_n$ are defined by the property that node k in preorder has e_k children and s_k descendants in the associated forest. (Thus s_k is the size of k's left subtree in the binary tree; also, s_k+1 is the length of the SCOPE link in the sense of 2.3.3–(5).) The next column repeats the fourteen forests of Table 1 in the lexicographic ordering of Algorithm P, but mirror-reversed from left to right.

Table 2

LINKED BINARY TREES AND RELATED OBJECTS WHEN $n = 4$

$l_1l_2l_3l_4$	$r_1r_2r_3r_4$	binary tree	forest	$e_1e_2e_3e_4$	$s_1s_2s_3s_4$	colex forest	lsib/rchild
2340	0000			1110	3210		
0340	2000			0110	0210		
2040	0300			2010	3010		
2040	3000			1010	1010		
0040	2300			0010	0010		
2300	0040			1200	3200		
0300	2040			0200	0200		
2300	0400			2100	3100		
2300	4000			1100	2100		
0300	2400			0100	0100		
2000	0340			3000	3000		
2000	4300			2000	2000		
2000	3040			1000	1000		
0000	2340			0000	0000		









And the final column shows the binary tree that represents the colex forest; it also happens to represent the forest in column 4, but by links to left sibling and right child instead of to left child and right sibling. This final column provides an interesting connection between nested parentheses and binary trees, so it gives us some insight into why Algorithm B is valid (see exercise 19).

***Gray codes for trees.** Our previous experiences with other combinatorial patterns suggest that we can probably generate parentheses and trees by making only small perturbations to get from one instance to another. And indeed, there are at least three very nice ways to achieve this goal.

Consider first the case of nested parentheses, which we can represent by the sequences $z_1z_2\ldots z_n$ that satisfy condition (8). A "near-perfect" way to generate all such combinations, in the sense of Section 7.2.1.3, is one in which we run through all possibilities in such a way that some component z_j changes by ±1 or ±2 at each step; this means that we get from each string of parentheses to its successor by simply changing either `()` $\leftrightarrow$ `)(` or `())` $\leftrightarrow$ `))(` in the vicinity of the jth left parenthesis. Here's one way to do the job when $n = 4$:

$$1357, 1356, 1346, 1345, 1347, 1247, 1245, 1246, 1236, 1234, 1235, 1237, 1257, 1256.$$

And we can extend any solution for $n-1$ to a solution for n, by taking each pattern $z_1z_2\ldots z_{n-1}$ and letting z_n run through all of its legal values using *endo-order* or its reverse as in 7.2.1.3–(45), proceeding downward from $2n-2$ and then up to $2n-1$ or vice versa, and omitting all elements that are $\le z_{n-1}$.

Algorithm N (*Near-perfect nested parentheses*). This algorithm visits all n-combinations $z_1\ldots z_n$ of $\{1,\ldots,2n\}$ that represent the indices of left parentheses in a nested string, changing only one index at a time. The process is controlled by an auxiliary array $g_1\ldots g_n$ that represents temporary goals.

N1. [Initialize.] Set $z_j \leftarrow 2j-1$ and $g_j \leftarrow 2j-2$ for $1 \le j \le n$.

N2. [Visit.] Visit the n-combination $z_1\ldots z_n$. Then set $j \leftarrow n$.

N3. [Find j.] If $z_j = g_j$, set $g_j \leftarrow g_j \oplus 1$ (thereby complementing the least significant bit), $j \leftarrow j-1$, and repeat this step.

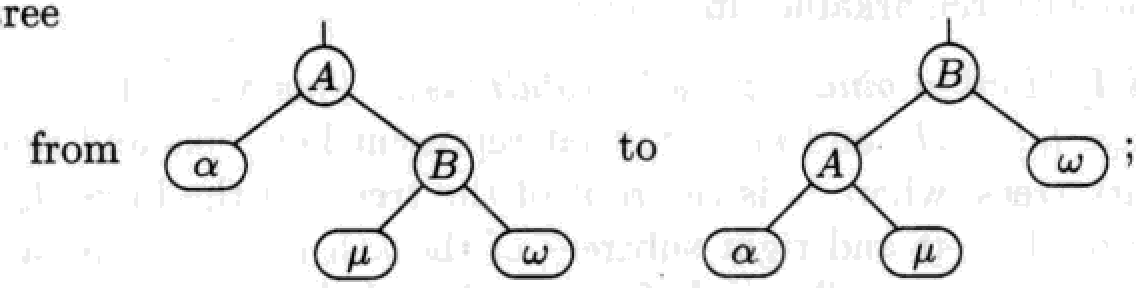
N4. [Home stretch?] If $g_j - z_j$ is even, set $z_j \leftarrow z_j + 2$ and return to N2.

N5. [Decrease or turn.] Set $t \leftarrow z_j - 2$. If $t < 0$, terminate the algorithm. Otherwise, if $t \le z_{j-1}$, set $t \leftarrow t + 2[t < z_{j-1}] + 1$. Finally set $z_j \leftarrow t$ and go back to N2. ▌

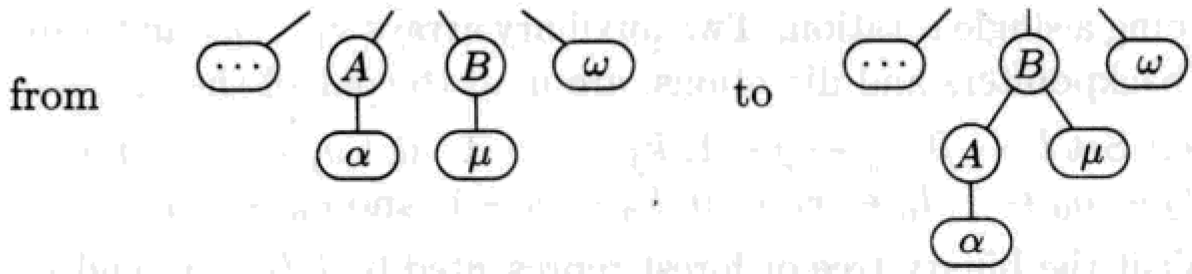
[A somewhat similar algorithm was introduced by D. Roelants van Baronaigien in *J. Algorithms* **35** (2000), 100–107; see also Xiang, Ushijima, and Tang, *Inf. Proc. Letters* **76** (2000), 169–174. F. Ruskey and A. Proskurowski, in *J. Algorithms* **11** (1990), 68–84, had previously shown how to construct *perfect* Gray codes for all tables $z_1\ldots z_n$ when $n \ge 4$ is even, thus changing some z_j by only ±1 at every step; but their construction was quite complex, and no known perfect scheme is simple enough to be of practical use. Exercise 48 shows that perfection is impossible when $n \ge 5$ is odd.]

If our goal is to generate linked tree structures instead of strings of parentheses, perfection of the z-index changes is not good enough, because simple swaps like `()` $\leftrightarrow$ `)(` don't necessarily correspond to simple link manipulations. A far better approach can be based on the "rotation" algorithms by which we were

able to keep search trees balanced in Section 6.2.3. *Rotation to the left* changes a binary tree


(12)

thus the corresponding forest is changed


(13)

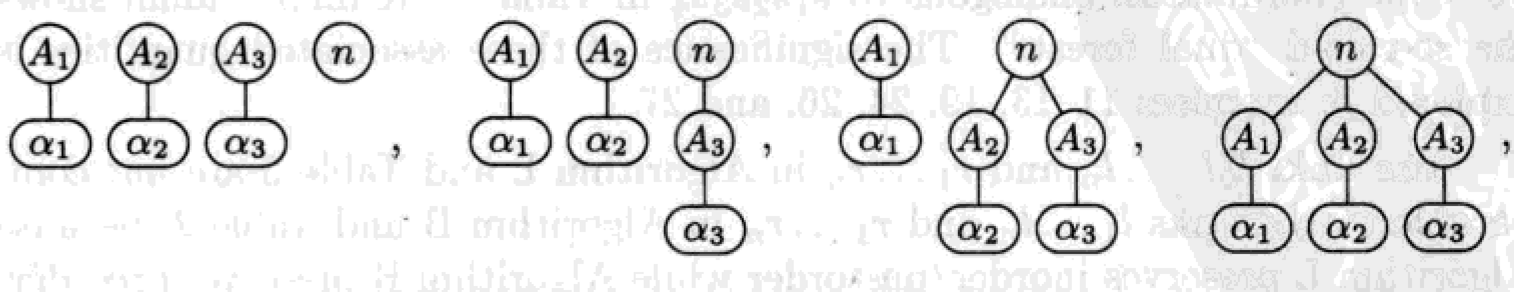
"Node (A) becomes the leftmost child of its right sibling." *Rotation to the right* is, of course, the opposite transformation: "The leftmost child of (B) becomes its left sibling." The vertical line in (12) stands for a connection to the overall context, either a left link or a right link or the pointer to the root. Any or all of the subtrees α, μ, or ω may be empty. The '$\cdots$' in (13), which represents additional siblings at the left of the family containing (B), might also be empty.

The nice thing about rotations is that only three links change: The right link from (A), the left link from (B), and the pointer from above. Rotations preserve inorder of the binary tree and postorder of the forest. (Notice also that the binary-tree form of a rotation corresponds in a natural way to an application of the *associative law*

$$(\alpha\mu)\omega = \alpha(\mu\omega) \qquad (14)$$

in the midst of an algebraic formula.)

A simple scheme very much like the classical reflected Gray code for n-tuples (Algorithm 7.2.1.1H) and the method of plain changes for permutations (Algorithm 7.2.1.2P) can be used to generate all binary trees or forests via rotations. Consider any forest on $n-1$ nodes, with k roots $(A_1), \ldots, (A_k)$. Then there are $k+1$ forests on n nodes that have the same postorder sequence on the first $n-1$ nodes but with node (n) last; for example, when $k = 3$ they are



obtained by successively rotating (A_3), (A_2), and (A_1) to the left. Moreover, at the extremes when (n) is either at the right or at the top, we can perform any desired rotation on the other $n-1$ nodes, because node (n) isn't in the way. Therefore, as observed by J. M. Lucas, D. Roelants van Baronaigien, and F. Ruskey [*J. Algorithms* **15** (1993), 343–366], we can extend any list of the $(n-1)$-node trees to a list of all n-node trees by simply letting node (n) roam

back and forth. A careful attention to low-level details makes it possible in fact to do the job with remarkable efficiency:

Algorithm L (*Linked binary trees by rotations*). This algorithm generates all pairs of arrays $l_0l_1\ldots l_n$ and $r_1\ldots r_n$ that represent left links and right links of n-node binary trees, where l_0 is the root of the tree and the links (l_k, r_k) point respectively to the left and right subtrees of the kth node in symmetric order. Equivalently, it generates all n-node forests, where l_k and r_k denote the left child and right sibling of the kth node in postorder. Each tree is obtained from its predecessor by doing a single rotation. Two auxiliary arrays $k_1\ldots k_n$ and $o_0o_1\ldots o_n$, representing backpointers and directions, are used to control the process.

L1. [Initialize.] Set $l_j \leftarrow 0$, $r_j \leftarrow j+1$, $k_j \leftarrow j-1$, and $o_j \leftarrow -1$ for $1 \le j < n$; also set $l_0 \leftarrow o_0 \leftarrow 1$, $l_n \leftarrow r_n \leftarrow 0$, $k_n \leftarrow n-1$, and $o_n \leftarrow -1$.

L2. [Visit.] Visit the binary tree or forest represented by $l_0l_1\ldots l_n$ and $r_1\ldots r_n$. Then set $j \leftarrow n$ and $p \leftarrow 0$.

L3. [Find j.] If $o_j > 0$, set $m \leftarrow l_j$ and go to L5 if $m \ne 0$. If $o_j < 0$, set $m \leftarrow k_j$; then go to L4 if $m \ne 0$, otherwise set $p \leftarrow j$. If $m = 0$ in either case, set $o_j \leftarrow -o_j$, $j \leftarrow j-1$, and repeat this step.

L4. [Rotate left.] Set $r_m \leftarrow l_j$, $l_j \leftarrow m$, $x \leftarrow k_m$, and $k_j \leftarrow x$. If $x = 0$, set $l_p \leftarrow j$, otherwise set $r_x \leftarrow j$. Return to L2.

L5. [Rotate right.] Terminate if $j = 0$. Otherwise set $l_j \leftarrow r_m$, $r_m \leftarrow j$, $k_j \leftarrow m$, $x \leftarrow k_m$. If $x = 0$, set $l_p \leftarrow m$, otherwise set $r_x \leftarrow m$. Go back to L2. ▮

Exercise 38 proves that Algorithm L needs only about 9 memory references per tree generated; thus it is almost as fast as Algorithm B. (In fact, two memory references per step could be saved by keeping the three quantities o_n, l_n, and k_n in registers. But of course Algorithm B can be speeded up too.)

Table 3 shows the sequence of binary trees and forests visited by Algorithm L when $n = 4$, with some auxiliary tables that shed further light on the process. The permutation $q_1q_2q_3q_4$ lists the nodes in preorder, when they have been numbered in postorder of the forest (symmetric order of the binary tree); it is the inverse of the permutation $p_1p_2p_3p_4$ in Table 1. The "coforest" is the conjugate (right-to-left reflection) of the forest; and the numbers $u_1u_2u_3u_4$ are its scope coordinates, analogous to $s_1s_2s_3s_4$ in Table 2. A final column shows the so-called "dual forest." The significance of these associated quantities is explored in exercises 11–13, 19, 24, 26, and 27.

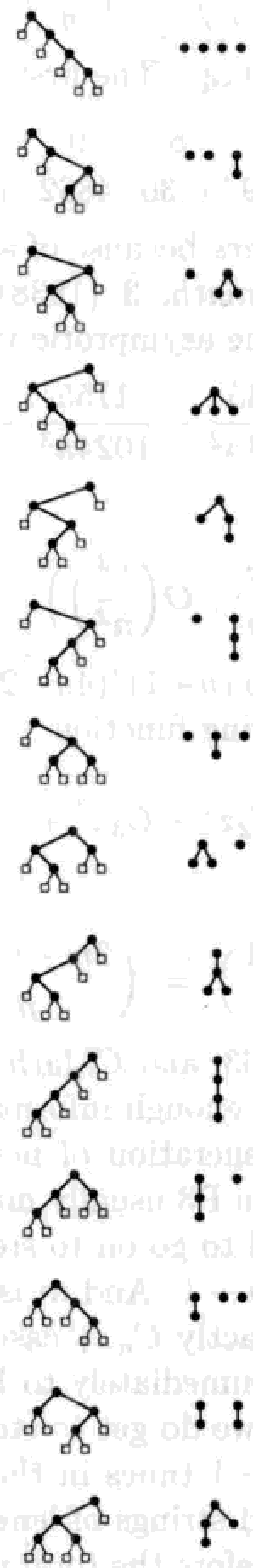
The links $l_0l_1\ldots l_n$ and $r_1\ldots r_n$ in Algorithm L and Table 3 are *not* comparable to the links $l_1\ldots l_n$ and $r_1\ldots r_n$ in Algorithm B and Table 2, because Algorithm L preserves inorder/postorder while Algorithm B preserves preorder. Node k in Algorithm L is the kth node from left to right in the binary tree, so l_0 is needed to identify the root; but node k in Algorithm B is the kth node in preorder, so the root is always node 1 in that case.

Algorithm L has the desired property that only three links change per step; but we can actually do even better in this respect if we stick to the preorder convention of Algorithm B. Exercise 25 presents an algorithm that generates

Table 3

BINARY TREES AND FORESTS GENERATED BY ROTATIONS WHEN $n = 4$

$l_0l_1l_2l_3l_4$	$r_1r_2r_3r_4$	$k_1k_2k_3k_4$	binary tree	forest	$q_1q_2q_3q_4$	coforest	$u_1u_2u_3u_4$	dual
10000	2340	0123			1234		0000	
10003	2400	0122			1243		1000	
10002	4300	0121			1423		2000	
40001	2300	0120			4123		3000	
40021	3000	0110			4132		3100	
10023	4000	0111			1432		2100	
10020	3040	0113			1324		0100	
30010	2040	0103			3124		0200	
40013	2000	0100			4312		3200	
40123	0000	0000			4321		3210	
30120	0040	0003			3214		0210	
20100	0340	0023			2134		0010	
20103	0400	0022			2143		1010	
40102	0300	0020			4213		3010	





all linked binary trees or forests by changing just two links per step, preserving preorder. One link becomes zero while another becomes nonzero. This prune-and-graft algorithm, which is the third of the three "very nice Gray codes for trees" promised above, has only one downside: Its controlling mechanism is a bit trickier than that of Algorithm L, so it needs about 40% more time to do the calculations when we include the cost of deciding what links to change at each step.

The number of trees. There's a simple formula for the total number of outputs that are generated by Algorithms P, B, N, and L, namely

$$C_n = \frac{1}{n+1}\binom{2n}{n} = \binom{2n}{n} - \binom{2n}{n-1}; \qquad (15)$$

we proved this fact in Eq. 2.3.4.4–(14). The first few values are

n =	0	1	2	3	4	5	6	7	8	9	10	11	12	13
C_n =	1	1	2	5	14	42	132	429	1430	4862	16796	58786	208012	742900

and they are called *Catalan numbers* because of some influential papers written by Eugène Catalan [*Journal de math.* **3** (1838), 508–515; **4** (1839), 95–99]. Stirling's approximation tells us the asymptotic value,

$$C_n = \frac{4^n}{\sqrt{\pi}\, n^{3/2}}\Bigl(1 - \frac{9}{8n} + \frac{145}{128n^2} - \frac{1155}{1024n^3} + \frac{36939}{32768n^4} + O(n^{-5})\Bigr); \qquad (16)$$

in particular we can conclude that

$$\frac{C_{n-k}}{C_n} = \frac{1}{4^k}\Bigl(1 + \frac{3k}{2n} + O\Bigl(\frac{k^2}{n^2}\Bigr)\Bigr) \qquad \text{when } |k| \le \frac{n}{2}. \qquad (17)$$

(And of course C_{n-1}/C_n is equal to $(n+1)/(4n-2)$, exactly, by (15).) In Section 2.3.4.4 we also derived the generating function

$$C(z) = C_0 + C_1z + C_2z^2 + C_3z^3 + \cdots = \frac{1-\sqrt{1-4z}}{2z} \qquad (18)$$

and proved the important formula

$$[z^n]\, C(z)^r = \frac{r}{n+r}\binom{2n+r-1}{n} = \binom{2n+r-1}{n} - \binom{2n+r-1}{n-1}; \qquad (19)$$

see the answer to exercise 2.3.4.4–33, and *CMath* equation (5.70).

These facts give us more than enough information to analyze Algorithm P, our algorithm for lexicographic generation of nested parentheses. Step P2 is obviously performed C_n times; then P3 usually makes a simple change and goes back to P2. How often do we need to go on to step P4? Easy: It's the number of times that step P2 finds $m = 2n-1$. And m is the location of the rightmost '(', so we have $m = 2n-1$ in exactly C_{n-1} cases. Thus the probability that P3 sets $m \leftarrow m-1$ and returns immediately to P2 is $(C_n - C_{n-1})/C_n \approx 3/4$, by (17). On the other hand when we do get to step P4, suppose we need to set $a_j \leftarrow$ ')' and $a_k \leftarrow$ '(' exactly $h-1$ times in that step. The number of cases with $h > x$ is the number of nested strings of length $2n$ that end with x trivial pairs () … (), namely C_{n-x}. Therefore the total number of times the algorithm changes a_j and a_k in step P4 is

$$\begin{aligned} C_{n-1} + C_{n-2} + \cdots + C_1 &= C_n\Bigl(\frac{C_{n-1}}{C_n} + \frac{C_{n-2}}{C_n} + \cdots + \frac{C_1}{C_n}\Bigr) \\ &= \frac{1}{3}C_n\Bigl(1 + \frac{2}{n} + O\Bigl(\frac{1}{n^2}\Bigr)\Bigr), \end{aligned} \qquad (20)$$

by (17); we have proved the claim for efficiency made earlier.

For a deeper understanding it is helpful to study the recursive structure underlying Algorithm P, as expressed in (5). The sequences A_{pq} in that formula have C_{pq} elements, where

$$C_{pq} = C_{p(q-1)} + C_{(p-1)q}, \quad \text{if } 0 \le p \le q \ne 0; \qquad C_{00} = 1; \tag{21}$$

and $C_{pq} = 0$ if $p < 0$ or $p > q$. Thus we can form the triangular array

$$\begin{matrix} C_{00} \\ C_{01} & C_{11} \\ C_{02} & C_{12} & C_{22} \\ C_{03} & C_{13} & C_{23} & C_{33} \\ C_{04} & C_{14} & C_{24} & C_{34} & C_{44} \\ C_{05} & C_{15} & C_{25} & C_{35} & C_{45} & C_{55} \\ C_{06} & C_{16} & C_{26} & C_{36} & C_{46} & C_{56} & C_{66} \end{matrix} \;=\; \begin{matrix} 1 \\ 1 & 1 \\ 1 & 2 & 2 \\ 1 & 3 & 5 & 5 \\ 1 & 4 & 9 & 14 & 14 \\ 1 & 5 & 14 & 28 & 42 & 42 \\ 1 & 6 & 20 & 48 & 90 & 132 & 132 \end{matrix} \tag{22}$$

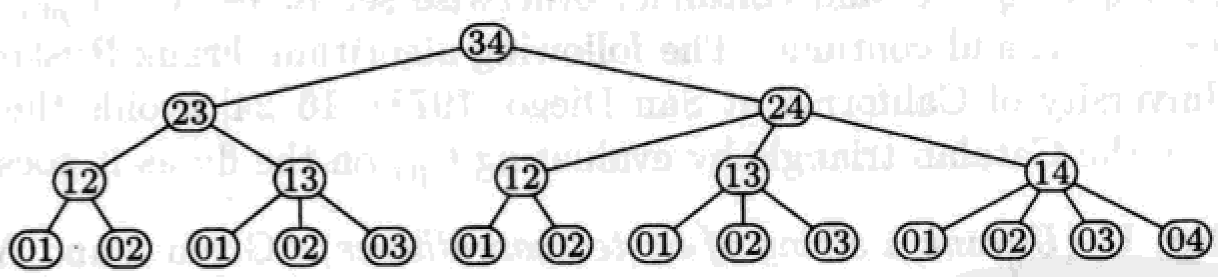
in which every entry is the sum of its nearest neighbors above and to the left; the Catalan numbers $C_n = C_{nn}$ appear on the diagonal. The elements of this triangle, which themselves have a venerable pedigree going back to de Moivre in 1711, are called "ballot numbers," because they represent sequences of $p+q$ ballots for which a running tabulation never favors a candidate with p votes over an opponent who receives q votes. The general formula

$$C_{pq} = \frac{q-p+1}{q+1}\binom{p+q}{p} = \binom{p+q}{p} - \binom{p+q}{p-1} \tag{23}$$

can be proved by induction or in a variety of more interesting ways; see exercise 39 and the answer to exercise 2.2.1–4. Notice that, because of (19), we have

$$C_{pq} = [z^p]\, C(z)^{q-p+1}. \tag{24}$$

When $n = 4$, Algorithm P essentially describes the recursion tree



(25)

because the specification (5) implies that $A_{nn} = (\,A_{(n-1)n}$ and that

$$A_{pq} = \,)^{q-p}(A_{(p-1)p},\ )^{q-p-1}(A_{(p-1)(p+1)},\ )^{q-p-2}(A_{(p-1)(p+2)},$$
$$\ldots,\ (A_{(p-1)q} \qquad \text{when } 0 \le p < q. \tag{26}$$

The number of leaves below node (pq) in this recursion tree is C_{pq}, and node (pq) appears exactly $C_{(n-q)(n-1-p)}$ times on level $n-1-p$; therefore we must have

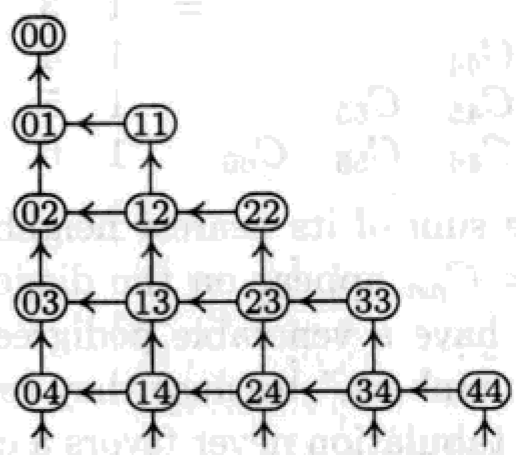
$$\sum_q C_{(n-q)(n-1-p)} C_{pq} = C_n, \qquad \text{for } 0 \le p < n. \tag{27}$$

The fourteen leaves of (25), from left to right, correspond to the fourteen rows of Table 1, from top to bottom. Notice that the entries in column $c_1c_2c_3c_4$ of that table assign the respective numbers 0000, 0001, 0010, ..., 0123 to the leaves

of (25), in accord with "Dewey decimal notation" for tree nodes (but with indices starting at 0 instead of 1, and with an extra 0 tacked on at the beginning).

A worm that crawls from one leaf to the next, around the bottom of the recursion tree, will ascend and descend h levels when h of the coordinates $c_1 \ldots c_n$ are changed, namely when Algorithm P resets the values of h '('s and h ')'s. This observation makes it easy to understand our previous conclusion that the condition $h > x$ occurs exactly C_{n-x} times during a complete crawl.

Yet another way to understand Algorithm P arises when we contemplate an infinite directed graph that is suggested by the recursion (21):



(28)

Clearly C_{pq} is the number of paths from ⓟⓠ to ⓪⓪ in this digraph, because of (21). And indeed, every string of parentheses in A_{pq} corresponds directly to such a path, with '(' signifying a step to the left and ')' signifying a step upward. Algorithm P explores all such paths systematically by trying first to go upward when extending a partial path.

Therefore it is easy to determine the Nth string of nested parentheses that is visited by Algorithm P, by starting at node (nn) and doing the following calculation when at node (pq): If $p = q = 0$, stop; otherwise, if $N \le C_{p(q-1)}$, emit ')', set $q \leftarrow q - 1$, and continue; otherwise set $N \leftarrow N - C_{p(q-1)}$, emit '(', set $p \leftarrow p - 1$, and continue. The following algorithm [Frank Ruskey, Ph.D. thesis (University of California at San Diego, 1978), 16–24] avoids the need to precompute the Catalan triangle by evaluating C_{pq} on the fly as it goes:

Algorithm U (*Unrank a string of nested parentheses*). Given n and N, where $1 \le N \le C_n$, this algorithm computes the Nth output $a_1 \ldots a_{2n}$ of Algorithm P.

U1. [Initialize.] Set $q \leftarrow n$ and $m \leftarrow p \leftarrow c \leftarrow 1$. While $p < n$, set $p \leftarrow p + 1$ and $c \leftarrow ((4p - 2)c)/(p + 1)$.

U2. [Done?] Terminate the algorithm if $q = 0$.

U3. [Go up?] Set $c' \leftarrow ((q + 1)(q - p)c)/((q + p)(q - p + 1))$. (At this point we have $1 \le N \le c = C_{pq}$ and $c' = C_{p(q-1)}$.) If $N \le c'$, set $q \leftarrow q - 1$, $c \leftarrow c'$, $a_m \leftarrow$ ')', $m \leftarrow m + 1$, and return to U2.

U4. [Go left.] Set $p \leftarrow p - 1$, $c \leftarrow c - c'$, $N \leftarrow N - c'$, $a_m \leftarrow$ '(', $m \leftarrow m + 1$, and return to U3. ▌

Random trees. We could choose a string $a_1 a_2 \ldots a_{2n}$ of nested parentheses at random by simply applying Algorithm U to a random integer N between 1

and C_n. But that idea isn't really very good, when n is bigger than 32 or so, because C_n can be quite large. A simpler and better way, proposed by D. B. Arnold and M. R. Sleep [*ACM Trans. Prog. Languages and Systems* **2** (1980), 122–128], is to generate a random "worm walk" by starting at (nn) in (28) and repeatedly taking leftward or upward branches with the appropriate probabilities. The resulting algorithm is almost the same as Algorithm U, but it deals only with nonnegative integers less than n^2+n+1:

Algorithm W (*Uniformly random strings of nested parentheses*). This algorithm generates a random string $a_1a_2\ldots a_{2n}$ of properly nested (s and)s.

W1. [Initialize.] Set $p \leftarrow q \leftarrow n$ and $m \leftarrow 1$.

W2. [Done?] Terminate the algorithm if $q=0$.

W3. [Go up?] Let X be a random integer in the range $0 \le X < (q+p)(q-p+1)$. If $X < (q+1)(q-p)$, set $q \leftarrow q-1$, $a_m \leftarrow$ ')', $m \leftarrow m+1$, and return to W2.

W4. [Go left.] Set $p \leftarrow p-1$, $a_m \leftarrow$ '(', $m \leftarrow m+1$, and return to W3. ▌

A worm's walk can be regarded as a sequence $w_0w_1\ldots w_{2n}$, where w_m is the worm's current depth after m steps. Thus, $w_0=0$; $w_m = w_{m-1}+1$ when $a_m =$ '('; $w_m = w_{m-1}-1$ when $a_m =$ ')'; and we have $w_m \ge 0$, $w_{2n}=0$. The sequence $w_0w_1\ldots w_{30}$ corresponding to (1) and (2) is 0121012321234345432321232343210. At step W3 of Algorithm W we have $q+p = 2n+1-m$ and $q-p = w_{m-1}$.

Let's say that the *outline* of a forest is the path that runs through the points $(m, -w_m)$ in the plane, for $0 \le m \le 2n$, where $w_0w_1\ldots w_{2n}$ is the worm walk corresponding to the associated string $a_1\ldots a_{2n}$ of nested parentheses. Figure 36 shows what happens if we plot the outlines of all 50-node forests and darken each point according to the number of forests that lie above it. For example, w_1 is always 1, so the triangular region at the upper left of Fig. 36 is solid black. But w_2 is either 0 or 2, and 0 occurs in $C_{49} \approx C_{50}/4$ cases; so the adjacent diamond-shaped area is a 75% shade of gray. Thus Fig. 36 illustrates the shape of a random forest, analogous to the shapes of random partitions that we've seen in Figs. 30, 31, and 35 of Sections 7.2.1.4 and 7.2.1.5.

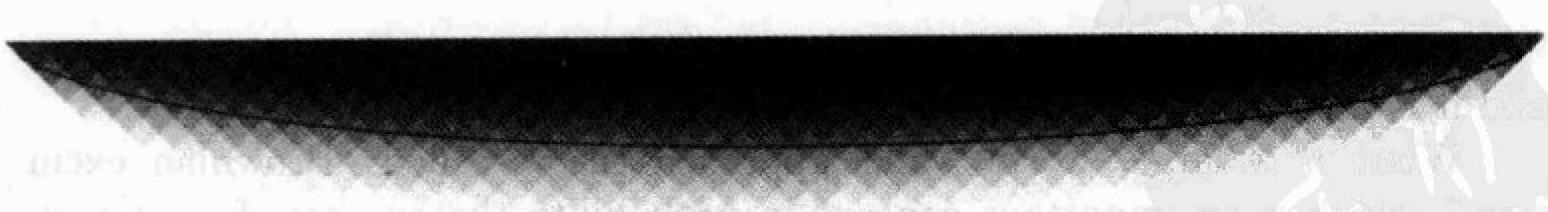


Fig. 36. The shape of a random 50-node forest.

Of course we can't really draw the outlines of all those forests, since there are C_{50} = 1,978,261,657,756,160,653,623,774,456 of them. But with the help of mathematics we can pretend that we've done so. The probability that $w_{2m}=2k$ is $C_{(m-k)(m+k)}C_{(n-m-k)(n-m+k)}/C_n$, because there are $C_{(m-k)(m+k)}$ ways to start with $m+k$ (s and $m-k$)s, and $C_{(n-m-k)(n-m+k)}$ ways to finish with

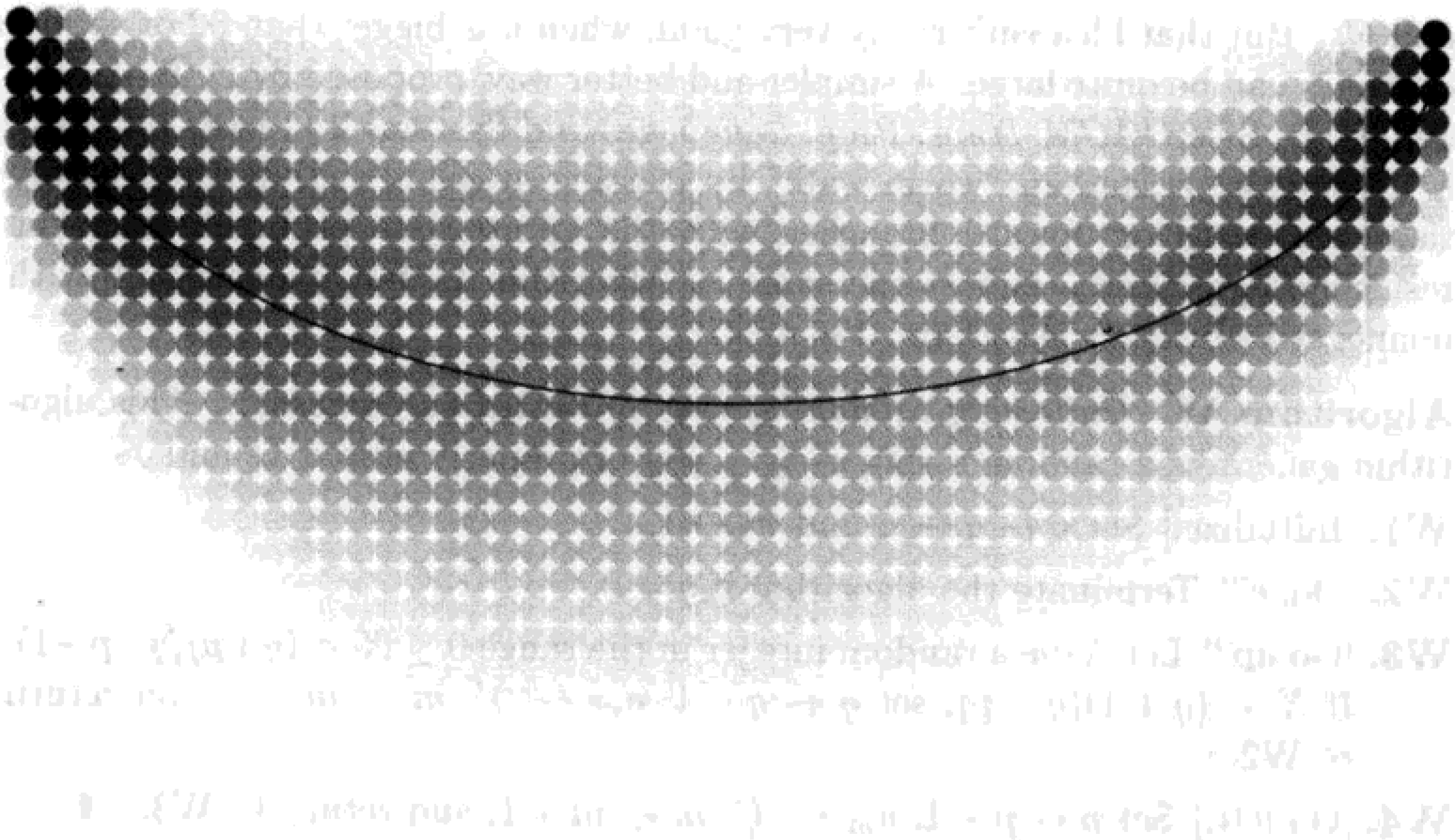


Fig. 37. Locations of the internal nodes in a random 50-node binary tree.

$n-(m+k)$ (s and $n-(m-k)$)s. By (23) and Stirling's approximation, this probability is

$$\frac{(2k+1)^2(n+1)}{(m+k+1)(n-m+k+1)}\binom{2m}{m-k}\binom{2n-2m}{n-m+k}\bigg/\binom{2n}{n}$$
$$=\frac{(2k+1)^2}{\sqrt{\pi}\,(\theta(1-\theta)n)^{3/2}}\,e^{-k^2/(\theta(1-\theta)n)}\Bigl(1+O\Bigl(\frac{k+1}{n}\Bigr)+O\Bigl(\frac{k^3}{n^2}\Bigr)\Bigr) \qquad (29)$$

when $m=\theta n$ and $n\to\infty$, for $0<\theta<1$. The average value of w_{2m} is worked out in exercise 57; it comes to

$$\frac{(4m(n-m)+n)\binom{2m}{m}\binom{2n-2m}{n-m}}{n\binom{2n}{n}}-1 = 4\sqrt{\frac{\theta(1-\theta)n}{\pi}}-1+O(n^{-1/2}), \qquad (30)$$

and it is illustrated for $n=50$ as a curved line in Fig. 36.

When n is large, worm walks approach the so-called "Brownian excursion," which is an important concept in probability theory. See, for example, Paul Lévy, *Processus Stochastiques et Mouvement Brownien* (1948), 225–237; Guy Louchard, *J. Applied Prob.* **21** (1984), 479–499, and *BIT* **26** (1986), 17–34; David Aldous, *Electronic Communications in Probability* **3** (1998), 79–90; Jon Warren, *Electronic Communications in Probability* **4** (1999), 25–29; J.-F. Marckert, *Random Structures and Algorithms* **24** (2004), 118–132.

What is the shape of a random *binary* tree? This question was investigated by Frank Ruskey in *SIAM J. Algebraic and Discrete Methods* **1** (1980), 43–50, and the answer turns out to be quite interesting. Suppose we draw a binary tree

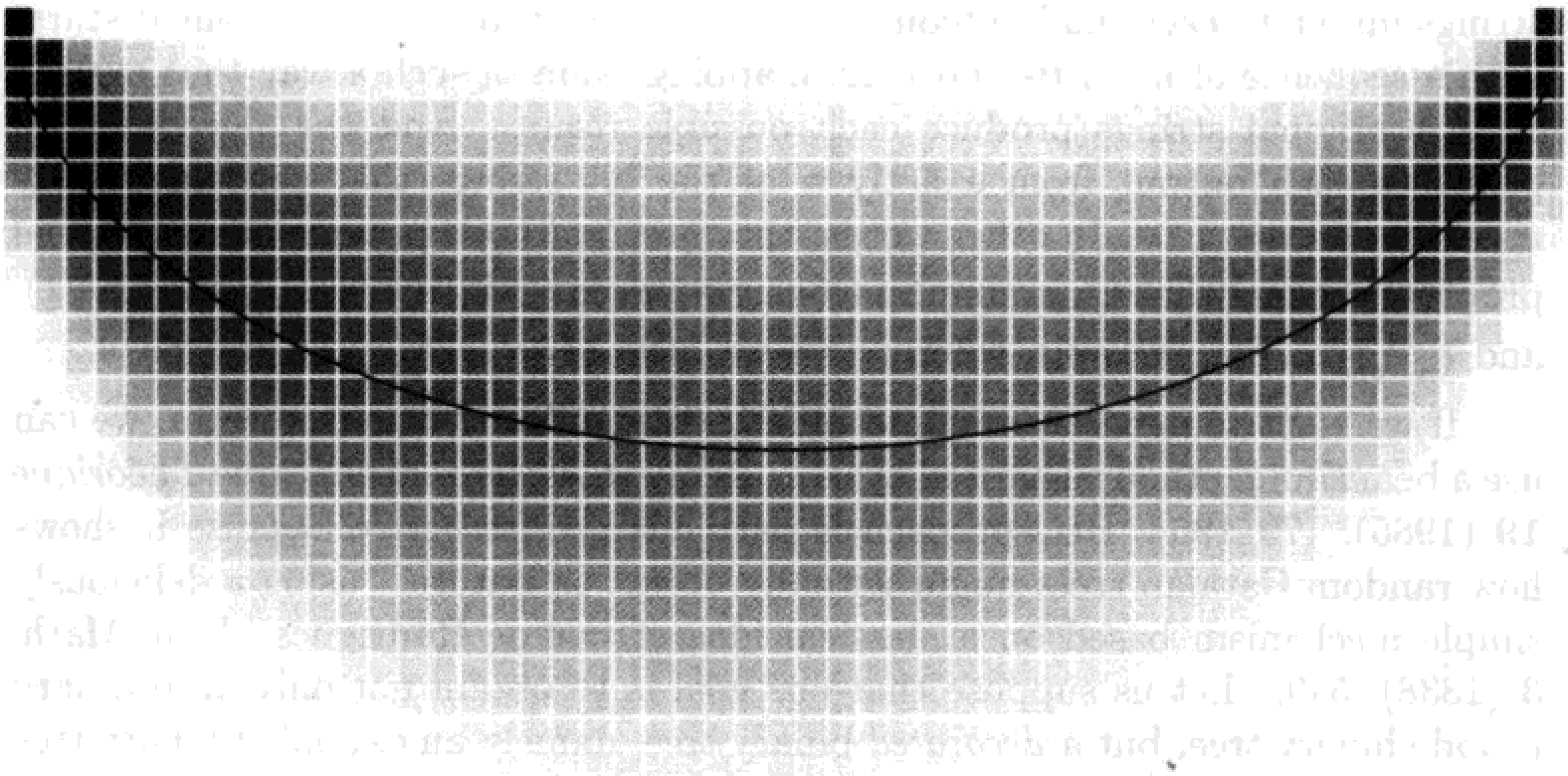


Fig. 38. Locations of the external nodes in a random 50-node binary tree.

as in (4), with the mth internal node at horizontal position m when the nodes are numbered in symmetric order. If all of the 50-node binary trees are drawn in this way and superimposed on each other, we get the distribution of node positions shown in Fig. 37. Similarly, if we number the *external* nodes from 0 to n in symmetric order and place them at horizontal positions .5, 1.5, …, $n+.5$, the "fringes" of all 50-node binary trees form the distribution shown in Fig. 38. Notice that the root node is most likely to be either number 1 or number n, at the extreme left or right; it is least likely to be either $\lfloor (n+1)/2 \rfloor$ or $\lceil (n+1)/2 \rceil$, in the middle.

As in Fig. 36, the smooth curves in Figs. 37 and 38 show the average node depths; exact formulas are derived in exercises 58 and 59. Asymptotically, the average depth of external node m is

$$8\sqrt{\frac{\theta(1-\theta)n}{\pi}} - 1 + O\Big(\frac{1}{\sqrt{n}}\Big), \qquad \text{when } m = \theta n \text{ and } n \to \infty, \tag{31}$$

for all fixed ratios θ with $0 < \theta < 1$, curiously like (30); and the average depth of *internal* node m is asymptotically the same, but with '-1' replaced by '-3'. Thus we can say that *the average shape of a random binary tree is approximately the lower half of an ellipse, n units wide and $4\sqrt{n/\pi}$ levels deep.*

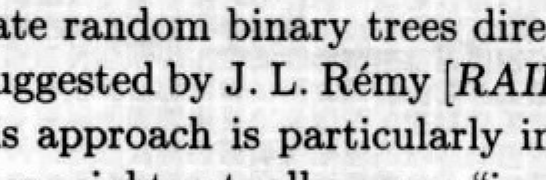
Three other noteworthy ways to generate random encodings of forests are discussed in exercises 60, 61, and 62. They are less direct than Algorithm W, yet they have substantial combinatorial interest. The first one begins with an arbitrary random string containing n (s and n) s, not necessarily nested; each of the $\binom{2n}{n}$ possibilities is equally likely. It then proceeds to convert every such string into a sequence that is properly nested, in such a way that exactly $n+1$

strings map into each final outcome. The second method is similar, but it starts with a sequence of $n+1$ 0s and n 2s, mapping them in such a way that exactly $2n+1$ original strings produce each possible result. And the third method produces each output from exactly n of the bit strings that contain exactly $n-1$ 1s and $n+1$ 0s. In other words, the three methods provide combinatorial proofs of the fact that C_n is simultaneously equal to $\binom{2n}{n}/(n+1)$, $\binom{2n+1}{n}/(2n+1)$, and $\binom{2n}{n-1}/n$. For example, when $n=4$ we have $14 = 70/5 = 126/9 = 56/4$.

If we want to generate random binary trees directly in linked form, we can use a beautiful method suggested by J. L. Rémy [*RAIRO Informatique Théorique* **19** (1985), 179–195]. His approach is particularly instructive because it shows how random Catalan trees might actually occur "in nature," using a deliciously simple mechanism based on a classical idea of Olinde Rodrigues [*J. de Math.* **3** (1838), 549]. Let us suppose that our goal is to obtain not only an ordinary n-node binary tree, but a *decorated* binary tree, namely an extended binary tree in which the external nodes have been labeled with the numbers 0 to n in some order. There are $(n+1)!$ ways to decorate any given binary tree; so the total number of decorated binary trees with n internal nodes is

$$D_n = (n+1)!\,C_n = \frac{(2n)!}{n!} = (4n-2)D_{n-1}. \qquad (32)$$

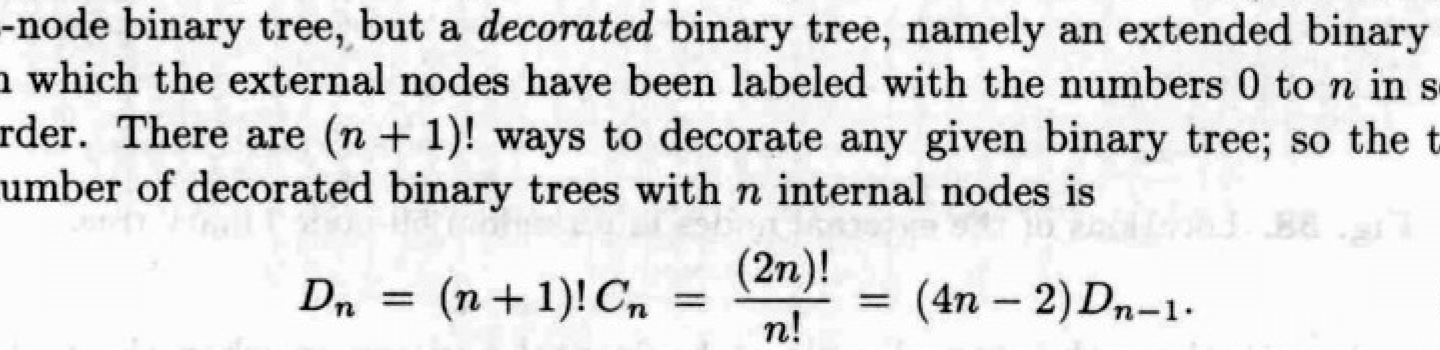
Rémy observed that there are $4n-2$ easy ways to build a decorated tree of order n from a given decorated tree of order $n-1$: We simply choose any one of the $2n-1$ nodes (internal or external) in the given tree, say x, and replace it by either



, (33)

thus inserting a new internal node and a new leaf while moving x and its descendants (if any) down one level.

For example, here's one way to construct a decorated tree of order 6:


(34)

Notice that every decorated tree is obtained by this process in exactly one way, because the predecessor of each tree must be the tree we get by striking out the highest-numbered leaf. Therefore Rémy's construction produces decorated trees that are uniformly random; and if we ignore the external nodes, we get random binary trees of the ordinary, undecorated variety.

One appealing way to implement Rémy's procedure is to maintain a table of links $L_0L_1\ldots L_{2n}$ where external (leaf) nodes have even numbers and internal (branch) nodes have odd numbers. The root is node L_0; the left and right children of branch node $2k-1$ are respectively L_{2k-1} and L_{2k}, for $1 \le k \le n$. Then the program is short and sweet:

Algorithm R (*Growing a random binary tree*). This algorithm constructs the linked representation $L_0L_1 \ldots L_{2N}$ of a uniformly random binary tree with N internal nodes, using the conventions explained above.

R1. [Initialize.] Set $n \leftarrow 0$ and $L_0 \leftarrow 0$.

R2. [Done?] (At this point the links $L_0L_1 \ldots L_{2n}$ represent a random n-node binary tree.) Terminate the algorithm if $n = N$.

R3. [Advance n.] Let X be a random integer between 0 and $4n+1$, inclusive. Set $n \leftarrow n+1$, $b \leftarrow X \bmod 2$, $k \leftarrow \lfloor X/2 \rfloor$, $L_{2n-b} \leftarrow 2n$, $L_{2n-1+b} \leftarrow L_k$, $L_k \leftarrow 2n-1$, and return to R2. ▮

***Chains of subsets.** Now that we've got trees and parentheses firmly in mind, it's a good time to discuss the *Christmas tree pattern*,* which is a remarkable way to arrange the set of all 2^n bit strings of length n into $\binom{n}{\lfloor n/2 \rfloor}$ rows and $n+1$ columns, discovered by de Bruijn, van Ebbenhorst Tengbergen, and Kruyswijk [*Nieuw Archief voor Wiskunde* (2) **23** (1951), 191–193].

The Christmas tree pattern of order 1 is the single row '0 1'; and the pattern of order 2 is

$$\begin{matrix} & 10 & \\ 00 & 01 & 11 \end{matrix} . \tag{35}$$

In general we get the Christmas tree pattern of order $n+1$ by taking every row '$\sigma_1\ \sigma_2\ \ldots\ \sigma_s$' of the order-$n$ pattern and replacing it by the two rows

$$\begin{matrix} & \sigma_2 0 & \ldots & \sigma_s 0 & \\ \sigma_1 0 & \sigma_1 1 & \ldots & \sigma_{s-1} 1 & \sigma_s 1 \end{matrix} . \tag{36}$$

(The first of these rows is omitted when $s = 1$.)

Proceeding in this way, we obtain for example the pattern of order 8 that appears in Table 4 on the next page. It is easy to verify by induction that

i) Each of the 2^n bit strings appears exactly once in the pattern.

ii) The bit strings with k 1s all appear in the same column.

iii) Within each row, consecutive bit strings differ by changing a 0 to a 1.

If we think of the bit strings as representing subsets of $\{1, \ldots, n\}$, with 1-bits to indicate the members of a set, property (iii) says that each row represents a *chain* in which each subset is covered by its successor. In symbols, using the notation of Section 7.1, every row $\sigma_1\ \sigma_2\ \ldots\ \sigma_s$ has the property that $\sigma_j \subseteq \sigma_{j+1}$ and $\nu(\sigma_{j+1}) = \nu(\sigma_j) + 1$ for $1 \le j < s$.

Properties (i) and (ii) tell us that there are exactly $\binom{n}{k}$ elements in column k, if we number the columns from 0 to n. This observation, together with the fact that each row is centered among the columns, proves that the total number of rows is $\max_{0 \le k \le n} \binom{n}{k} = \binom{n}{\lfloor n/2 \rfloor}$, as claimed. Let us call this number M_n.

---

* This name was chosen for sentimental reasons, because the pattern has a general shape not unlike that of a festive tree, and because it was the subject of the author's ninth annual "Christmas Tree Lecture" at Stanford University in December 2002.

Table 4

THE CHRISTMAS TREE PATTERN OF ORDER 8

				10101010				
			10101000	10101001	10101011			
				10101100				
			10100100	10100101	10101101			
			10100010	10100110	10101110			
		10100000	10100001	10100011	10100111	10101111		
				10110010				
			10110000	10110001	10110011			
				10110100				
			10010100	10010101	10110101			
			10010010	10010110	10110110			
		10010000	10010001	10010011	10010111	10110111		
				10111000				
			10011000	10011001	10111001			
			10001010	10011010	10111010			
		10001000	10001001	10001011	10011011	10111011		
			10001100	10011100	10111100			
		10000100	10000101	10001101	10011101	10111101		
		10000010	10000110	10001110	10011110	10111110		
	10000000	10000001	10000011	10000111	10001111	10011111	10111111	
				11001010				
			11001000	11001001	11001011			
				11001100				
			11000100	11000101	11001101			
			11000010	11000110	11001110			
		11000000	11000001	11000011	11000111	11001111		
				11010010				
			11010000	11010001	11010011			
				11010100				
			01010100	01010101	11010101			
			01010010	01010110	11010110			
		01010000	01010001	01010011	01010111	11010111		
				11011000				
			01011000	01011001	11011001			
			01001010	01011010	11011010			
		01001000	01001001	01001011	01011011	11011011		
			01001100	01011100	11011100			
		01000100	01000101	01001101	01011101	11011101		
		01000010	01000110	01001110	01011110	11011110		
	01000000	01000001	01000011	01000111	01001111	01011111	11011111	
				11100010				
			11100000	11100001	11100011			
				11100100				
			01100100	01100101	11100101			
			01100010	01100110	11100110			
		01100000	01100001	01100011	01100111	11100111		
				11101000				
			01101000	01101001	11101001			
			00101010	01101010	11101010			
		00101000	00101001	00101011	01101011	11101011		
			00101100	01101100	11101100			
		00100100	00100101	00101101	01101101	11101101		
		00100010	00100110	00101110	01101110	11101110		
	00100000	00100001	00100011	00100111	00101111	01101111	11101111	
				11110000				
			01110000	01110001	11110001			
			00110010	01110010	11110010			
		00110000	00110001	00110011	01110011	11110011		
			00110100	01110100	11110100			
		00010100	00010101	00110101	01110101	11110101		
		00010010	00010110	00110110	01110110	11110110		
	00010000	00010001	00010011	00010111	00110111	01110111	11110111	
			00111000	01111000	11111000			
		00011000	00011001	00111001	01111001	11111001		
		00001010	00011010	00111010	01111010	11111010		
	00001000	00001001	00001011	00011011	00111011	01111011	11111011	
		00001100	00011100	00111100	01111100	11111100		
	00000100	00000101	00001101	00011101	00111101	01111101	11111101	
	00000010	00000110	00001110	00011110	00111110	01111110	11111110	
00000000	00000001	00000011	00000111	00001111	00011111	00111111	01111111	11111111

A set C of bit strings is called a *clutter*, or an "antichain of subsets," if its bit strings are incomparable in the sense that $\sigma \not\subseteq \tau$ whenever σ and τ are distinct elements of C. A famous theorem of Emanuel Sperner [*Math. Zeitschrift* **27** (1928), 544–548] asserts that no clutter on $\{1,\ldots,n\}$ can have more than M_n elements; and the Christmas tree pattern provides a simple proof, because no clutter can contain more than one element of each row.

Indeed, the Christmas tree pattern can be used to show that much more is true. Let's note first that exactly $\binom{n}{k} - \binom{n}{k-1}$ rows of length $n+1-2k$ are present, for $0 \le k \le n/2$, because there are exactly $\binom{n}{k}$ elements in column k. For example, Table 4 has one row of length 9, namely the bottom row; it also has $\binom{8}{1} - \binom{8}{0} = 7$ rows of length 7, $\binom{8}{2} - \binom{8}{1} = 20$ rows of length 5, $\binom{8}{3} - \binom{8}{2} = 28$ of length 3, and $\binom{8}{4} - \binom{8}{3} = 14$ of length 1. Moreover, these numbers $\binom{n}{k} - \binom{n}{k-1}$ appear in the Catalan triangle (22), because they're equal to $C_{k(n-k)}$ according to Eq. (23).

Further study reveals that this Catalan connection is not simply a coincidence; nested parentheses are, in fact, the key to a deeper understanding of the Christmas tree pattern, because the theory of parentheses tells us where an arbitrary bit string fits into the array. Suppose we use the symbols (and) instead of 1 and 0, respectively. Any string of parentheses, nested or not, can be written uniquely in the form

$$\alpha_0\,)\,\ldots\,\alpha_{p-1}\,)\,\alpha_p\,(\,\alpha_{p+1}\,\ldots\,(\,\alpha_q \tag{37}$$

for some p and q with $0 \le p \le q$, where the substrings $\alpha_0, \ldots, \alpha_q$ are properly nested and possibly empty; exactly p of the right parentheses and $q-p$ of the left parentheses are "free" in the sense that they have no mate. For example, the string

$$)\,(\,(\,)\,)\,(\,)\,)\,(\,)\,)\,)\,)\,(\,(\,(\,(\,(\,)\,(\,(\,)\,(\,)\,(\,(\,(\,)\,) \tag{38}$$

has $p=5$, $q=12$, $\alpha_0=\epsilon$, $\alpha_1 = (())()$, $\alpha_2 = ()$, $\alpha_3=\epsilon$, $\ldots$, $\alpha_{12} = (())$. In general, the string (37) is part of a chain of length $q+1$,

$$\alpha_0\,)\ldots\alpha_{q-1}\,)\,\alpha_q,\;\; \alpha_0\,)\ldots\alpha_{q-2}\,)\,\alpha_{q-1}\,(\,\alpha_q,\;\; \ldots,\;\; \alpha_0\,(\,\alpha_1\ldots(\,\alpha_q, \tag{39}$$

in which we start with q free)s and change them one by one into free (s. Every row of the Christmas tree pattern is obtained in exactly this manner, but using 1 and 0 instead of (and); for if the chain $\sigma_1 \ldots \sigma_s$ corresponds to the nested strings $\alpha_0, \ldots, \alpha_{s-1}$, its successor chains in (36) correspond respectively to $\alpha_0, \ldots, \alpha_{s-3}, \alpha_{s-2}(\alpha_{s-1})$ and to $\alpha_0, \ldots, \alpha_{s-3}, \alpha_{s-2}, \alpha_{s-1}, \epsilon$. [See Curtis Greene and Daniel J. Kleitman, *J. Combinatorial Theory* **A20** (1976), 80–88.]

Notice furthermore that the rightmost elements in each row of the pattern—such as 10101010, 10101011, 10101100, 10101101, ..., 11111110, 11111111 in the case $n=8$—are in lexicographic order. Thus, for example, the fourteen rows of length 1 in Table 4 correspond precisely to the fourteen strings of nested parentheses in Table 1. This observation makes it easy to generate the rows of Table 8 sequentially from bottom to top, with a method analogous to Algorithm P; see exercise 77.

Let $f(x_1, \ldots, x_n)$ be any monotone Boolean function of n variables. If $\sigma = a_1 \ldots a_n$ is any bit string of length n, we can write $f(\sigma) = f(a_1, \ldots, a_n)$ for convenience. Any row $\sigma_1 \ldots \sigma_s$ of the Christmas tree pattern forms a chain, so we have

$$0 \le f(\sigma_1) \le \cdots \le f(\sigma_s) \le 1. \tag{40}$$

In other words, there is an index t such that $f(\sigma_j) = 0$ for $j < t$ and $f(\sigma_j) = 1$ for $j \ge t$; we will know the value of $f(\sigma)$ for all 2^n bit strings σ if we know the indices t for each row of the pattern.

Georges Hansel [*Comptes Rendus Acad. Sci.* (A) **262** (Paris, 1966), 1088–1090] noticed that the Christmas tree pattern has another important property: If σ_{j-1}, σ_j, and σ_{j+1} are three consecutive entries of any row, the bit string

$$\sigma'_j = \sigma_{j-1} \oplus \sigma_j \oplus \sigma_{j+1} \tag{41}$$

lies in a *previous* row. In fact, σ'_j lies in the same column as σ_j, and it satisfies

$$\sigma_{j-1} \subseteq \sigma'_j \subseteq \sigma_{j+1}; \tag{42}$$

it is called the relative complement of σ_j in the interval $(\sigma_{j-1} \,..\, \sigma_{j+1})$. Hansel's observation is easy to prove by induction, because of the recursive rule (36) that defines the Christmas tree pattern. He used it to show that we can deduce the values of $f(\sigma)$ for all σ by actually evaluating the function at relatively few well-chosen places; for if we know the value of $f(\sigma'_j)$, we will know either $f(\sigma_{j-1})$ or $f(\sigma_{j+1})$ because of relation (42).

Algorithm H (*Learning a monotone Boolean function*). Let $f(x_1, \ldots, x_n)$ be a Boolean function that is nondecreasing in each Boolean variable, but otherwise unknown. Given a bit string σ of length n, let $r(\sigma)$ be the number of the row in which σ appears in the Christmas tree pattern, where $1 \le r(\sigma) \le M_n$. If $1 \le m \le M_n$, let $s(m)$ be the number of bit strings in row m; also let $\chi(m, k)$ be the bit string in column k of that row, for $(n+1-s(m))/2 \le k \le (n-1+s(m))/2$. This algorithm determines the sequence of threshold values $t(1)$, $t(2)$, $\ldots$, $t(M_n)$ such that

$$f(\sigma) = 1 \iff \nu(\sigma) \ge t(r(\sigma)), \tag{43}$$

by evaluating f at no more than two points per row.

H1. [Loop on m.] Perform steps H2 through H4 for $m = 1, \ldots, M_n$; then stop.

H2. [Begin row m.] Set $a \leftarrow (n + 1 - s(m))/2$ and $z \leftarrow (n - 1 + s(m))/2$.

H3. [Do a binary search.] If $z \le a + 1$, go to H4. Otherwise set $k \leftarrow \lfloor (a+z)/2 \rfloor$, and

$$\sigma \leftarrow \chi(m, k-1) \oplus \chi(m, k) \oplus \chi(m, k+1). \tag{44}$$

If $k \ge t(r(\sigma))$, set $z \leftarrow k$; otherwise set $a \leftarrow k$. Repeat step H3.

H4. [Evaluate.] If $f(\chi(m, a)) = 1$, set $t(m) \leftarrow a$; otherwise, if $a = z$, set $t(m) \leftarrow a + 1$; otherwise set $t(m) \leftarrow z + 1 - f(\chi(m, z))$. ▮

Hansel's algorithm is *optimum*, in the sense that it evaluates f at the fewest possible points in the worst case. For if f happens to be the threshold function

$$f(\sigma) = [\nu(\sigma) > n/2], \tag{45}$$

any valid algorithm that learns f on the first m rows of the Christmas tree pattern must evaluate $f(\sigma)$ in column $\lfloor n/2 \rfloor$ of each row, and in column $\lfloor n/2 \rfloor + 1$ of each row that has size greater than 1. Otherwise we could not distinguish f from a function that differs from it only at an unexamined point. [See V. K. Korobkov, *Problemy Kibernetiki* **13** (1965), 5–28, Theorem 5.]

Oriented trees and forests. Let's turn now to another kind of tree, in which the parent-child relationship is important but the order of children in each family is not. An *oriented forest* of n nodes can be defined by a sequence of pointers $p_1 \ldots p_n$, where p_j is the parent of node j (or $p_j = 0$ if j is a root); the directed graph on vertices $\{0, 1, \ldots, n\}$ with arcs $\{j \to p_j \mid 1 \le j \le n\}$ will have no oriented cycles. An *oriented tree* is an oriented forest with exactly one root. (See Section 2.3.4.2.) Every n-node oriented forest is equivalent to an $(n+1)$-node oriented tree, because the root of that tree can be regarded as the parent of all the roots of the forest. We saw in Section 2.3.4.4 that there are A_n oriented trees with n nodes, where the first few values are

$$\begin{matrix} n = 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 \\ A_n = 1 & 1 & 2 & 4 & 9 & 20 & 48 & 115 & 286 & 719 & 1842 & 4766 & 12486 & 32973 \end{matrix}\;; \tag{46}$$

asymptotically, $A_n = c\alpha^n n^{-3/2} + O(\alpha^n n^{-5/2})$ where $\alpha \approx 2.9558$ and $c \approx 0.4399$. Thus, for example, only 9 of the 14 forests in Table 1 are distinct when we ignore the horizontal left-to-right ordering and consider only the vertical orientation.

Every oriented forest corresponds to a unique ordered forest if we sort the members of each family appropriately, using an ordering on trees introduced by H. I. Scoins [*Machine Intelligence* **3** (1968), 43–60]: Recall from (11) that ordered forests can be characterized by their level codes $c_1 \ldots c_n$, where node j in preorder appears on level c_j. An ordered forest is called *canonical* if the level code sequences for the subtrees in each family are in nonincreasing lexicographic order. For example, the canonical forests in Table 1 are those whose level codes $c_1c_2c_3c_4$ are 0000, 0100, 0101, 0110, 0111, 0120, 0121, 0122, and 0123. The level sequence 0112 is not canonical, because the subtrees of the root have respective level codes 1 and 12; the string 1 is lexicographically less than 12. We can readily verify by induction that *the canonical level codes are lexicographically largest, among all ways of reordering the subtrees of a given oriented forest.*

T. Beyer and S. M. Hedetniemi [*SICOMP* **9** (1980), 706–712] noticed that there is a remarkably simple way to generate oriented forests if we visit them in *decreasing* lexicographic order of the canonical level codes. Suppose $c_1 \ldots c_n$ is canonical, where $c_k > 0$ and $c_{k+1} = \cdots = c_n = 0$. The next smallest sequence is obtained by decreasing c_k, then increasing $c_{k+1} \ldots c_n$ to the largest levels consistent with canonicity; and those levels are easy to compute. For if $j = p_k$ is the parent of node k, we have $c_j = c_k - 1 < c_l$ for $j < l \le k$, hence the levels $c_j \ldots c_k$

represent the subtree currently rooted at node j. To get the largest sequence of levels less than $c_1 \dots c_n$ we therefore replace $c_k \dots c_n$ by the first $n+1-k$ elements of the infinite sequence $(c_j \dots c_{k-1})^\infty = c_j \dots c_{k-1} c_j \dots c_{k-1} c_j \dots$. (The effect is to remove k from its current position as the rightmost child of j, then to append new subtrees that are siblings of j, by cloning j and its descendants as often as possible. This cloning process may terminate in the midst of the sequence $c_j \dots c_{k-1}$, but that causes no difficulty because every prefix of a canonical level sequence is canonical.) For example, to obtain the successor of any sequence of canonical codes that ends with 23443433000000000, we replace the final 3000000000 by 2344343234.

Algorithm O (*Oriented forests*). This algorithm generates all oriented forests on n nodes, by visiting all canonical n-node forests in decreasing lexicographic order of their level codes $c_1 \dots c_n$. The level codes are not computed explicitly, however; each canonical forest is represented directly by its sequence of parent pointers $p_1 \dots p_n$, in preorder of the nodes. To generate all oriented trees on $n+1$ nodes, we can imagine that node 0 is the root.

O1. [Initialize.] Set $p_k \leftarrow k-1$ for $0 \le k \le n$. (In particular, this step makes p_0 nonzero, for use in termination testing; see step O4.)

O2. [Visit.] Visit the forest represented by parent pointers $p_1 \dots p_n$.

O3. [Easy case?] If $p_n > 0$, set $p_n \leftarrow p_{p_n}$ and return to step O2.

O4. [Find j and k.] Find the largest $k < n$ such that $p_k \ne 0$. Terminate the algorithm if $k = 0$; otherwise set $j \leftarrow p_k$ and $d \leftarrow k - j$.

O5. [Clone.] If $p_{k-d} = p_j$, set $p_k \leftarrow p_j$; otherwise set $p_k \leftarrow p_{k-d} + d$. Return to step O2 if $k = n$; otherwise set $k \leftarrow k+1$ and repeat this step. ▌

As in other algorithms we've been seeing, the loops in steps O4 and O5 tend to be quite short; see exercise 88. Exercise 90 proves that slight changes to this algorithm suffice to generate all arrangements of edges that form *free* trees.

Spanning trees. Now let's consider the minimal subgraphs that "span" a given graph. If G is a connected graph on n vertices, the *spanning trees* of G are the subsets of $n-1$ edges that contain no cycles; equivalently, they are the subsets of edges that form a free tree connecting all the vertices. Spanning trees are important in many applications, especially in the study of networks, so the problem of generating all spanning trees has been treated by many authors. In fact, systematic ways to list them all were developed early in the 20th century by Wilhelm Feussner [*Annalen der Physik* (4) **9** (1902), 1304–1329], long before anybody thought about generating other kinds of trees.

In the following discussion we will allow graphs to have any number of edges between two vertices; but we disallow loops from a vertex to itself, because self-loops cannot be part of a tree. Feussner's basic idea was very simple, yet eminently suited for calculation: If e is any edge of G, a spanning tree either contains e or it doesn't. Suppose e joins vertex u to vertex v, and suppose it is part of a spanning tree; then the other $n-2$ edges of that tree span the graph

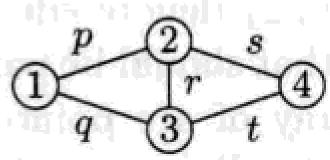
G / e that we obtain by regarding u and v as identical. In other words, the spanning trees that contain e are essentially the same as the spanning trees of the contracted graph G / e that results when we shrink e down to a single point. On the other hand the spanning trees that do *not* contain e are spanning trees of the reduced graph $G \setminus e$ that results when we eliminate edge e. Symbolically, therefore, the set $S(G)$ of all spanning trees of G satisfies

$$S(G) = e\,S(G / e) \cup S(G \setminus e). \qquad (47)$$

Malcolm J. Smith, in his Master's thesis at the University of Victoria (1999), introduced a nice way to carry out the recursion (47) by finding all spanning trees in a "revolving-door Gray code" order: Each tree in his scheme is obtained from its predecessor by simply removing one edge and substituting another. Such orderings are not difficult to find, but the trick is to do the job efficiently.

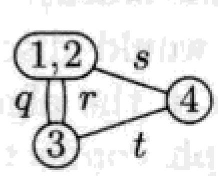
The basic idea of Smith's algorithm is to generate $S(G)$ in such a way that the first spanning tree includes a given *near tree*, namely a set of $n-2$ edges containing no cycle. This task is trivial if $n = 2$; we simply list all the edges. If $n > 2$ and if the given near tree is $\{e_1, \ldots, e_{n-2}\}$, we proceed as follows: Assume that G is connected; otherwise there are no spanning trees. Form G / e_1 and append e_1 to each of its spanning trees, beginning with one that contains $\{e_2, \ldots, e_{n-2}\}$; notice that $\{e_2, \ldots, e_{n-2}\}$ is a near tree of G/e_1, so this recursion makes sense. If the last spanning tree found in this way for G / e_1 is $f_1 \ldots f_{n-2}$, complete the task by listing all spanning trees for $G \setminus e_1$, beginning with one that contains the near tree $\{f_1, \ldots, f_{n-2}\}$.

For example, suppose G is the graph

$G =$ 

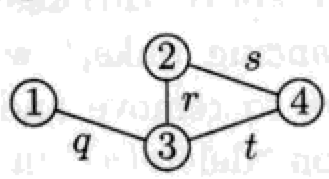
(48)

with four vertices and five edges $\{p, q, r, s, t\}$. Starting with the near tree $\{p, q\}$, Smith's procedure first forms the contracted graph

$G / p =$ 

(49)

and lists its spanning trees, beginning with one that contains q. This list might be *qs*, *qt*, *ts*, *tr*, *rs*; thus the trees *pqs*, *pqt*, *pts*, *ptr*, and *prs* span G. The remaining task is to list the spanning trees of

$G \setminus p =$ 

(50)

starting with one that contains $\{r, s\}$; they are *rsq*, *rqt*, *qts*.

A detailed implementation of Smith's algorithm turns out to be quite instructive. As usual we represent the graph by letting two arcs $u \to v$ and $v \to u$ correspond to each edge $u — v$, and we maintain lists of "arc nodes" to represent the arcs that leave each vertex. We'll need to shrink and unshrink the graph's

edges, so we will make these lists doubly linked. If a points to an arc node that represents $u \to v$, then

> $a \oplus 1$ points to the "mate" of a, which represents $v \to u$;
> t_a is the "tip" of a, namely v (hence $t_{a\oplus 1} = u$);
> i_a is an optional name that identifies this edge (and equals $i_{a\oplus 1}$);
> n_a points to the next element of u's arc list;
> p_a points to the previous element of u's arc list;

and l_a is a link used for undeleting arcs as explained below.

The vertices are represented by integers $\{1, \ldots, n\}$; and arc number $v - 1$ is a header node for vertex v's doubly linked arc list. A header node a is recognizable by the fact that its tip, t_a, is 0. We let d_v be the degree of vertex v. Thus, for example, the graph (48) might be represented by $(d_1, d_2, d_3, d_4) = (2, 3, 3, 2)$ and by the following fourteen nodes of arc data:

$a =$	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$t_a =$	0	0	0	0	1	2	1	3	2	3	2	4	3	4
$i_a =$					p	p	q	q	r	r	s	s	t	t
$n_a =$	5	4	6	10	9	7	8	0	13	11	12	1	3	2
$p_a =$	7	11	13	12	1	0	2	5	6	4	3	9	10	8

The implicit recursion of Smith's algorithm can be controlled conveniently by using an array of arc pointers $a_1 \ldots a_{n-1}$. At level l of the process, arcs $a_1 \ldots a_{l-1}$ denote edges that have been included in the current spanning tree; a_l is ignored; and arcs $a_{l+1} \ldots a_{n-1}$ denote edges of a near tree on the contracted graph $(\ldots(G/a_1)\ldots)/a_{l-1}$ that should be part of the next spanning tree visited.

There's also another array of arc pointers $s_1 \ldots s_{n-2}$, representing stacks of arcs that have been temporarily removed from the current graph. The top element of the stack for level l is s_l, and each arc a links to its successor, l_a (which is 0 at the bottom of the stack).

An edge whose removal would disconnect a connected graph is called a *bridge*. One of the key points in the algorithm that follows is the fact that we want to keep the current graph connected; therefore we don't set $G \leftarrow G \setminus e$ when e is a bridge.

Algorithm S (*All spanning trees*). Given a connected graph represented with the data structures explained above, this algorithm visits all of its spanning trees.

A technique called "dancing links," which we will discuss extensively in Section 7.2.2.1, is used here to remove and restore items from and to doubly linked lists. The abbreviation "delete(a)" in the steps below is shorthand for the pair of operations

$$n_{p_a} \leftarrow n_a, \quad p_{n_a} \leftarrow p_a; \tag{51}$$

similarly, "undelete(a)" stands for

$$p_{n_a} \leftarrow a, \quad n_{p_a} \leftarrow a. \tag{52}$$

S1. [Initialize.] Set $a_1 \ldots a_{n-1}$ to a spanning tree of the graph. (See exercise 94.) Also set $x \leftarrow 0$, $l \leftarrow 1$, and $s_1 \leftarrow 0$. If $n = 2$, set $v \leftarrow 1$, $e \leftarrow n_0$, and go to S5.

S2. [Enter level l.] Set $e \leftarrow a_{l+1}$, $u \leftarrow t_e$, and $v \leftarrow t_{e\oplus 1}$. If $d_u > d_v$, interchange $v \leftrightarrow u$ and set $e \leftarrow e \oplus 1$.

S3. [Shrink e.] (Now we will make u identical to v by inserting u's adjacency list into v's. We also must delete all former edges between u and v, including e itself, because such edges would otherwise become loops. Deleted edges are linked together so that we can restore them later in step S7.) Set $k \leftarrow d_u + d_v$, $f \leftarrow n_{u-1}$, and $g \leftarrow 0$. While $t_f \neq 0$, do the following: If $t_f = v$, delete(f), delete($f \oplus 1$), and set $k \leftarrow k - 2$, $l_f \leftarrow g$, $g \leftarrow f$; otherwise set $t_{f\oplus 1} \leftarrow v$. Then set $f \leftarrow n_f$ and repeat these operations until $t_f = 0$. Finally set $l_e \leftarrow g$, $d_v \leftarrow k$, $g \leftarrow v - 1$, $n_{p_f} \leftarrow n_g$, $p_{n_g} \leftarrow p_f$, $p_{n_f} \leftarrow g$, $n_g \leftarrow n_f$, and $a_l \leftarrow e$.

S4. [Advance l.] Set $l \leftarrow l + 1$. If $l < n - 1$, set $s_l \leftarrow 0$ and return to S2. Otherwise set $e \leftarrow n_{v-1}$.

S5. [Visit.] (The current graph now has only two vertices, one of which is v.) Set $a_{n-1} \leftarrow e$ and visit the spanning tree $a_1 \ldots a_{n-1}$. (If $x = 0$, this is the first spanning tree to be visited; otherwise it differs from its predecessor by deleting x and inserting e.) Set $x \leftarrow e$ and $e \leftarrow n_e$. Repeat step S5 if $t_e \neq 0$.

S6. [Decrease l.] Set $l \leftarrow l - 1$. Terminate the algorithm if $l = 0$; otherwise set $e \leftarrow a_l$, $u \leftarrow t_e$, and $v \leftarrow t_{e\oplus 1}$.

S7. [Unshrink e.] Set $f \leftarrow u - 1$, $g \leftarrow v - 1$, $n_g \leftarrow n_{p_f}$, $p_{n_g} \leftarrow g$, $n_{p_f} \leftarrow f$, $p_{n_f} \leftarrow f$, and $f \leftarrow p_f$. While $t_f \neq 0$, set $t_{f\oplus 1} \leftarrow u$ and $f \leftarrow p_f$. Then set $f \leftarrow l_e$, $k \leftarrow d_v$; while $f \neq 0$ set $k \leftarrow k + 2$, undelete($f \oplus 1$), undelete(f), and set $f \leftarrow l_f$. Finally set $d_v \leftarrow k - d_u$.

S8. [Test for bridge.] If e is a bridge, go to S9. (See exercise 95 for one way to perform this test.) Otherwise set $x \leftarrow e$, $l_e \leftarrow s_l$, $s_l \leftarrow e$; delete(e) and delete($e \oplus 1$). Set $d_u \leftarrow d_u - 1$, $d_v \leftarrow d_v - 1$, and go to S2.

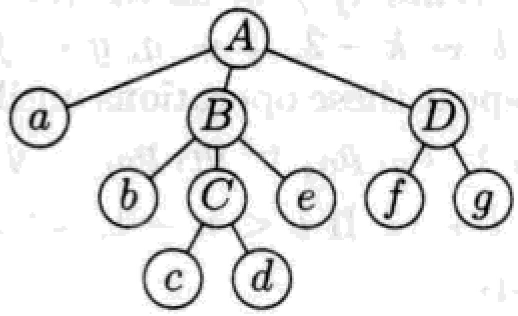
S9. [Undo level l deletions.] Set $e \leftarrow s_l$. While $e \neq 0$, set $u \leftarrow t_e$, $v \leftarrow t_{e\oplus 1}$, $d_u \leftarrow d_u + 1$, $d_v \leftarrow d_v + 1$, undelete($e \oplus 1$), undelete(e), and $e \leftarrow l_e$. Return to S6. ▮

The reader is encouraged to play through the steps of this algorithm on a small graph such as (48). Notice that a subtle case arises in steps S3 and S7, if u's adjacency list happens to become empty. Notice also that several shortcuts would be possible, at the expense of a more complicated algorithm; we will discuss such improvements later in this section.

***Series-parallel graphs.** The task of finding all spanning trees becomes especially simple when the given graph has a serial and/or parallel decomposition. A *series-parallel graph between s and t* is a graph G with two designated vertices, s and t, whose edges can be built up recursively as follows: Either G consists of a single edge, $s — t$; or G is a *serial superedge* consisting of $k \geq 2$ series-parallel subgraphs G_j between s_j and t_j, joined in series with $s = s_1$ and $t_j = s_{j+1}$ for

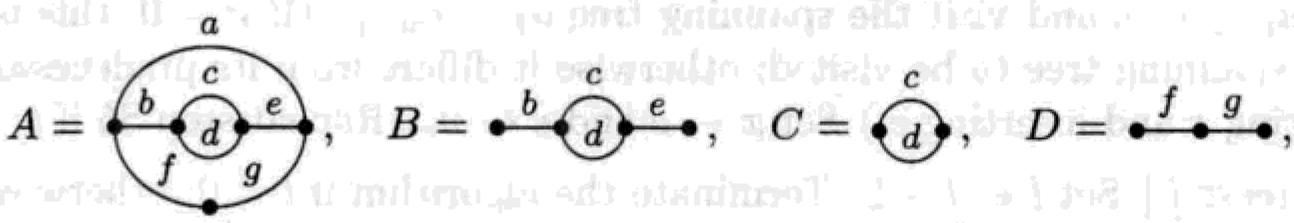
$1 \le j < k$ and $t_k = t$; or G is a *parallel superedge* consisting of $k \ge 2$ series-parallel subgraphs G_j between s and t joined in parallel. This decomposition is essentially unique, given s and t, if we require that the subgraphs G_j for serial superedges are not themselves serial superedges, and that the subgraphs G_j for parallel superedges are not themselves parallel.

Any series-parallel graph can be represented conveniently as a tree, with no nodes of degree 1. The leaf nodes of this tree represent edges, and the branch nodes represent superedges, alternating between serial and parallel from level to level. For example, the tree



(53)

corresponds to the series-parallel graphs and subgraphs



(54)

if the top node A is taken to be parallel. Edges are named in (54), but not vertices, because edges are of prime importance with respect to spanning trees.

Let's say that a *near tree* of a series-parallel graph between s and t is a set of $n-2$ cycle-free edges that do not connect s to t. The spanning trees and near trees of a series-parallel graph are easy to describe recursively, as follows: (1) A spanning tree of a serial superedge corresponds to spanning trees of all its principal subgraphs G_j; a near tree corresponds to spanning trees in all but one of the G_j, and a near tree in the other. (2) A near tree of a parallel superedge corresponds to near trees of all its principal subgraphs G_j; a spanning tree corresponds to near trees in all but one of the G_j, and a spanning tree in the other.

Rules (1) and (2) suggest the following data structures for listing the spanning trees and/or near trees of series-parallel graphs. Let p point to a node in a tree like (53). Then we define

$t_p = 1$ for serial superedges, 0 otherwise (the "type" of p);
$v_p = 1$ if we have a spanning tree for p, 0 if we have a near tree;
$l_p =$ pointer to p's leftmost child, or 0 if p is a leaf;
$r_p =$ pointer to p's right sibling, wrapping around cyclically;
$d_p =$ pointer to a designated child of p, or 0 if p is a leaf.

If q points to the rightmost child of p, its "right sibling" r_q equals l_p. And if q points to *any* child of p, rules (1) and (2) state that

$$v_q = \begin{cases} v_p, & \text{if } q = d_p; \\ t_p, & \text{if } q \ne d_p. \end{cases} \qquad (55)$$

(For example, if p is a branch node that represents a serial superedge, we must have $v_q = 1$ for all but one of p's children; the only exception is the designated child d_p. Thus we must have a spanning tree for all of the subgraphs that were joined serially to form p, except for one designated subgraph in the case that we have a near tree for p.)

Given any setting of the designated-child pointers d_p, and given any value 0 or 1 for v_p at the root of the tree, Eq. (55) tells us how to propagate values down to all of the leaves. For example, if we set $v_A \leftarrow 1$ in the tree (53), and if we designate the leftmost child of each branch node (so that $d_A = a$, $d_B = b$, $d_C = c$, and $d_D = f$), we find successively

$$v_a = 1,\ v_B = 0,\ v_b = 0,\ v_C = 1,\ v_c = 1,\ v_d = 0,\ v_e = 1,\ v_D = 0,\ v_f = 0,\ v_g = 1. \quad (56)$$

A leaf node q is present in the spanning tree if and only if $v_q = 1$; hence (56) specifies the spanning tree $aceg$ of the series-parallel graph A in (54).

For convenience, let's say that the *configs* of p are its spanning trees if $v_p = 1$, its near trees if $v_p = 0$. We would like to generate all configs of the root. A branch node p is called "easy" if $v_p = t_p$; that is, a serial node is easy if its configs are spanning trees, and a parallel node is easy if its configs are near trees. If p is easy, its configs are the Cartesian product of the configs of its children, namely all k-tuples of the children's configs, varying independently; the designated child d_p is immaterial in the easy case. But if p is uneasy, its configs are the union of such Cartesian k-tuples, taken over all possible choices of d_p.

As luck would have it, easy nodes are relatively rare: At most one child of an uneasy node (namely the designated child) can be easy, and all children of an easy node are uneasy unless they are leaves.

Even so, the tree representation of a series-parallel graph makes the recursive generation of all its spanning trees and/or near trees quite straightforward and efficient. The operations of Algorithm S — shrinking and unshrinking, deleting and undeleting, bridge detection — are not needed when we deal with series-parallel graphs. Furthermore, exercise 99 shows that there is a pleasant way to obtain the spanning trees or near trees in a revolving-door Gray code order, by using focus pointers as in several algorithms that we've seen earlier.

***Refinements of Algorithm S.** Although Algorithm S provides us with a simple and reasonably effective way to visit all spanning trees of a general graph, its author Malcolm Smith realized that the properties of series-parallel graphs can be used to make it even better. For example, if a graph has two or more edges that run between the same vertices u and v, we can combine them into a superedge; the spanning trees of the original graph can then be obtained readily from those of the simpler, reduced graph. And if a graph has a vertex v of degree 2, so that the only edges touching v are $u — v$ and $v — w$, we can eliminate v and replace those edges by a single superedge between u and w. Furthermore, any vertex of degree 1 can effectively be eliminated, together with its adjacent edge, by simply including that edge in every spanning tree.

After the reductions in the preceding paragraph have been applied to a given graph G, we obtain a reduced graph $\hat{G}$ having no parallel edges and no vertices

of degrees 1 or 2, together with a set of $m \geq 0$ series-parallel graphs $S_1, \ldots, S_m$, representing edges (or superedges) that must be included in all spanning trees of G. Every remaining edge $u — v$ of $\hat{G}$ corresponds, in fact, to a series-parallel graph S_{uv} between vertices u and v. *The spanning trees of G are then obtained as the union, taken over all spanning trees T of $\hat{G}$, of the Cartesian product of the spanning trees of $S_1, \ldots, S_m$ and the spanning trees of all S_{uv} for edges $u — v$ in T, together with the near trees of all S_{uv} for edges $u — v$ that are in $\hat{G}$ but not in T.* And all spanning trees T of $\hat{G}$ can be obtained by using the strategy of Algorithm S.

In fact, when Algorithm S is extended in this way, its operations of replacing the current graph G by G / e or $G \setminus e$ typically trigger further reductions, as new parallel edges appear or as the degree of a vertex drops below 3. Therefore it turns out that the "stopping state" of the implicit recursion in Algorithm S, namely the case when only two vertices are left (step S5), never actually arises: A reduced graph $\hat{G}$ either has only a single vertex and no edges, or it has at least four vertices and six edges.

The resulting algorithm retains the desirable revolving-door property of Algorithm S, and it is quite pretty (although about four times as long as the original); see exercise 100. Smith proved that it has the best possible asymptotic running time: If G has n vertices, m edges, and N spanning trees, the algorithm visits them all in $O(m + n + N)$ steps.

The performance of Algorithm S and of its souped-up version Algorithm S′ can best be appreciated by considering the number of memory accesses that those algorithms actually make when they generate the spanning trees of typical graphs, as shown in Table 5. The bottom line of that table corresponds to the graph *plane_miles*(16, 0, 0, 1, 0, 0, 0) from the Stanford GraphBase, which serves as an "organic" antidote to the purely mathematical examples on the previous lines. The random multigraph on the penultimate line, also from the Stanford GraphBase, can be described more precisely by its official name *random_graph*(16, 37, 1, 0, 0, 0, 0, 0, 0, 0). Although the 4×4 torus is isomorphic to the 4-cube (see exercise 7.2.1.1–17), those isomorphic graphs yield slightly different running times because their vertices and edges are encountered differently when the algorithms are run.

In general we can say that Algorithm S is not too bad on small examples, except when the graph is quite sparse; but Algorithm S′ begins to shine when many spanning trees are present. Once Algorithm S′ gets warmed up, it tends to crank out a new tree after every 18 or 19 mems go by.

Table 5 also indicates that a mathematically-defined graph often has a surprisingly "round" number of spanning trees. For example, D. M. Cvetković [*Srpska Akademija Nauka, Matematicheski Institut* **11** (Belgrade: 1971), 135–141] discovered, among other things, that the n-cube has exactly

$$2^{2^n-n-1}\, 1^{\binom{n}{1}}\, 2^{\binom{n}{2}} \ldots n^{\binom{n}{n}} \qquad (57)$$

of them. Exercises 104–109 explore some of the reasons why that happens.

Table 5

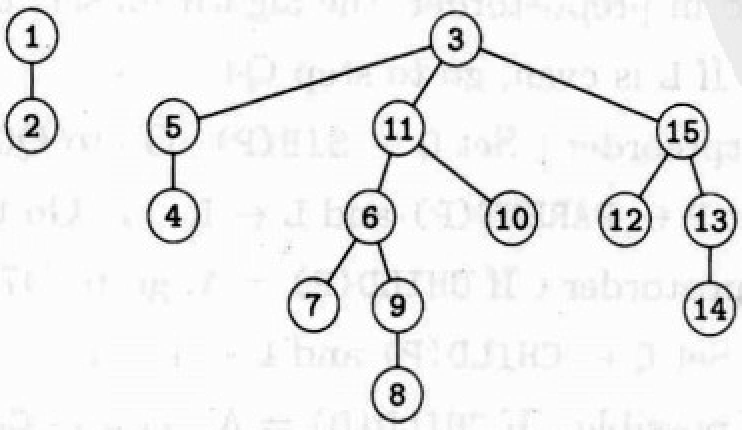
RUNNING TIME IN MEMS NEEDED TO GENERATE ALL SPANNING TREES

	m	n	N	Algorithm S	Algorithm S′	μ per tree	
path P_{10}	9	10	1	794 μ	473 μ	794.0	473.0
path P_{100}	99	100	1	9,974 μ	5,063 μ	9974.0	5063.0
cycle C_{10}	10	10	10	3,480 μ	998 μ	348.0	99.8
cycle C_{100}	100	100	100	355,605 μ	10,538 μ	3556.1	105.4
complete graph K_4	6	4	16	1,213 μ	1,336 μ	75.8	83.5
complete graph K_{10}	45	10	100,000,000	3,759.58 Mμ	1,860.95 Mμ	37.6	18.6
complete bigraph $K_{5,5}$	25	10	390,625	23.43 Mμ	8.88 Mμ	60.0	22.7
4×4 grid $P_4 \times P_4$	24	16	100,352	12.01 Mμ	1.87 Mμ	119.7	18.7
5×5 grid $P_5 \times P_5$	40	25	557,568,000	54.68 Gμ	10.20 Gμ	98.1	18.3
4×4 cylinder $P_4 \times C_4$	28	16	2,558,976	230.96 Mμ	49.09 Kμ	90.3	19.2
5×5 cylinder $P_5 \times C_5$	45	25	38,720,000,000	3,165.31 Gμ	711.69 Gμ	81.7	18.4
4×4 torus $C_4 \times C_4$	32	16	42,467,328	3,168.15 Mμ	823.08 Mμ	74.6	19.4
4-cube $P_2 \times P_2 \times P_2 \times P_2$	32	16	42,467,328	3,168.16 Mμ	823.38 Mμ	74.7	19.4
random multigraph	37	16	59,933,756	3,818.19 Mμ	995.91 Mμ	63.7	16.6
16 cities	37	16	179,678,881	11,772.11 Mμ	3,267.43 Mμ	65.5	18.2

A general quasi-Gray code. Let's close this section by discussing something completely different, yet still related to trees. Consider the following hybrid variants of the two standard ways to traverse a nonempty forest:

Prepostorder traversal	Postpreorder traversal
Visit the root of the first tree	Traverse the subtrees of the first tree, in prepostorder
Traverse the subtrees of the first tree, in postpreorder	Visit the root of the first tree
Traverse the remaining trees, in prepostorder	Traverse the remaining trees, in postpreorder

In the first case, every tree of the forest is traversed in prepostorder, with its root first; but the subtrees of those roots are traversed in postpreorder, with roots coming last. The second variant is similar but with 'pre' and 'post' interchanged. And in general, prepostorder visits roots first on every even-numbered level of the forest, but visits them last on the odd-numbered levels. For example, the forest in (2) becomes



(58)

when we label its nodes in prepostorder.

Prepostorder and postpreorder are not merely curiosities; they're actually useful. The reason is that adjacent nodes, in either of these orders, are always near each other in the forest. For example, nodes k and $k+1$ are adjacent in (58) for $k = 1, 4, 6, 8, 10, 13$; they are separated by only one node when $k = 3, 12, 14$; and they're three steps apart when $k = 2, 5, 7, 9, 11$ (if we imagine an invisible super-parent at the top of the forest). A moment's thought proves inductively that at most two nodes can possibly intervene between prepostorder neighbors or postpreorder neighbors — because postpreorder(F) always begins with the root of the first tree or its leftmost child, and prepostorder(F) always ends with the root of the last tree or its rightmost child.

Suppose we want to generate all combinatorial patterns of some kind, and we want to visit them in a Gray-code-like manner so that consecutive patterns are always "close" to each other. We can form, at least conceptually, the graph of all possible patterns p, with edges $p \mathrel{—} q$ for all pairs of patterns that are close to each other. The following theorem, due to Milan Sekanina [*Spisy Přírodovědecké Fakulty University v Brně*, No. 412 (1960), 137–142], proves that a pretty good Gray code is always possible, provided only that we can get from any pattern to any other in a sequence of short steps:

Theorem S. *The vertices of any connected graph can be listed in a cyclic order $(v_0, v_1, \ldots, v_{n-1})$ so that the distance between v_k and $v_{(k+1) \bmod n}$ is at most 3, for $0 \le k < n$.*

Proof. Find a spanning tree in the graph, and traverse it in prepostorder. ▌

Graph theorists traditionally say that the kth power of a graph G is the graph G^k whose vertices are those of G, with $u \mathrel{—} v$ in G^k if and only if there's a path of length k or less from u to v in G. Thus they can state Theorem S much more succinctly, when $n > 2$: *The cube of a connected graph is Hamiltonian.*

Prepostorder traversal is also useful when we want to visit the nodes of a tree in loopless fashion, with a bounded number of steps between stops:

Algorithm Q (*Prepostorder successor in a triply linked forest*). If `P` points to a node in a forest represented by links `PARENT`, `CHILD`, and `SIB`, corresponding to each node's parent, leftmost child, and right sibling, this algorithm computes `P`'s successor node, `Q`, in prepostorder. We assume that we know the level `L` at which `P` appears in the forest; the value of `L` is updated to be the level of `Q`. If `P` happens to be the final node in prepostorder, the algorithm sets `Q` $\leftarrow \Lambda$ and `L` $\leftarrow -1$.

Q1. [Pre or post?] If `L` is even, go to step Q4.

Q2. [Continue postpreorder.] Set `Q` $\leftarrow$ `SIB(P)`. Go to Q6 if `Q` $\neq \Lambda$.

Q3. [Move up.] Set `P` $\leftarrow$ `PARENT(P)` and `L` $\leftarrow$ `L` $- 1$. Go to Q7.

Q4. [Continue prepostorder.] If `CHILD(P)` $= \Lambda$, go to Q7.

Q5. [Move down.] Set `Q` $\leftarrow$ `CHILD(P)` and `L` $\leftarrow$ `L` $+ 1$.

Q6. [Move down if possible.] If `CHILD(Q)` $\neq \Lambda$, set `Q` $\leftarrow$ `CHILD(Q)` and `L` $\leftarrow$ `L` $+ 1$. Terminate the algorithm.

Q7. [Move right or up.] If SIB(P) $\neq \Lambda$, set Q $\leftarrow$ SIB(P); otherwise set Q $\leftarrow$ PARENT(P) and L $\leftarrow$ L $-$ 1. Terminate the algorithm. ▮

Notice that, as in Algorithm 2.4C, the link PARENT(P) is examined only if SIB(P) $= \Lambda$. A complete traversal is really a worm walk around the forest, like (3): The worm "sees" the nodes on even-numbered levels when it passes them on the left, and it sees the odd-level nodes when it passes them on the right.

EXERCISES

1. [*15*] If a worm crawls around the binary tree (4), how could it easily reconstruct the parentheses of (1)?

2. [*20*] (S. Zaks, 1980.) Modify Algorithm P so that it produces the combinations $z_1z_2\dots z_n$ of (8) instead of the parenthesis strings $a_1a_2\dots a_{2n}$.

▶ **3.** [*23*] Prove that (11) converts $z_1z_2\dots z_n$ to the inversion table $c_1c_2\dots c_n$.

4. [*20*] True or false: If the strings $a_1\dots a_{2n}$ are generated in lexicographic order, so are the corresponding strings $d_1\dots d_n$, $z_1\dots z_n$, $p_1\dots p_n$, and $c_1\dots c_n$.

5. [*15*] What tables $d_1\dots d_n$, $z_1\dots z_n$, $p_1\dots p_n$, and $c_1\dots c_n$ correspond to the nested parenthesis string (1)?

▶ **6.** [*20*] What *matching* corresponds to (1)? (See the final column of Table 1.)

7. [*16*] (a) What is the state of the string $a_1a_2\dots a_{2n}$ when Algorithm P terminates? (b) What do the arrays $l_1l_2\dots l_n$ and $r_1r_2\dots r_n$ contain when Algorithm B terminates?

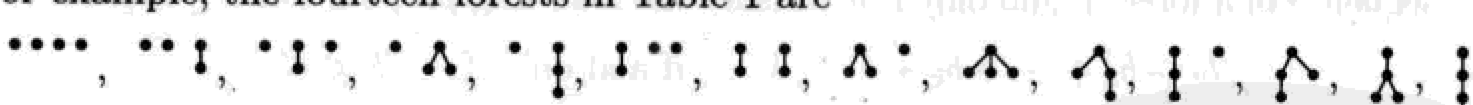
8. [*15*] What tables $l_1\dots l_n$, $r_1\dots r_n$, $e_1\dots e_n$, and $s_1\dots s_n$ correspond to the example forest (2)?

9. [*M20*] Show that the tables $c_1\dots c_n$ and $s_1\dots s_n$ are related by the law

$$c_k = [s_1 \geq k-1] + [s_2 \geq k-2] + \cdots + [s_{k-1} \geq 1].$$

10. [*M20*] (*Worm walks.*) Given a string of nested parentheses $a_1a_2\dots a_{2n}$, let w_j be the excess of left parentheses over right parentheses in $a_1a_2\dots a_j$, for $0 \leq j \leq 2n$. Prove that $w_0 + w_1 + \cdots + w_{2n} = 2(c_1 + \cdots + c_n) + n$.

11. [*11*] If F is a forest, its *conjugate* F^R is obtained by left-to-right mirror reflection. For example, the fourteen forests in Table 1 are

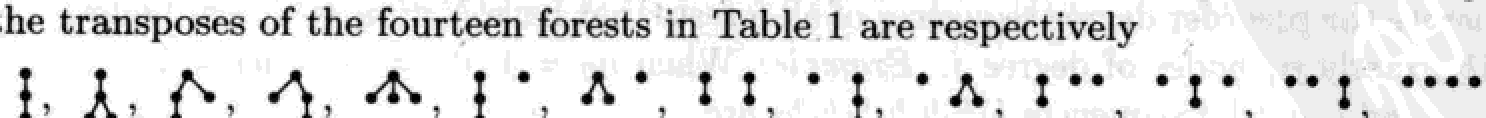


and their conjugates are respectively



as in the colex forests of Table 2. If F corresponds to the nested parentheses $a_1a_2\dots a_{2n}$, what string of parentheses corresponds to F^R?

12. [*15*] If F is a forest, its *transpose* F^T is the forest whose binary tree is obtained by interchanging left and right links in the binary tree representing F. For example, the transposes of the fourteen forests in Table 1 are respectively



What is the transpose of the forest (2)?

13. [*20*] Continuing exercises 11 and 12, how do the preorder and postorder of a labeled forest F relate to the preorder and postorder of (a) F^R? (b) F^T?

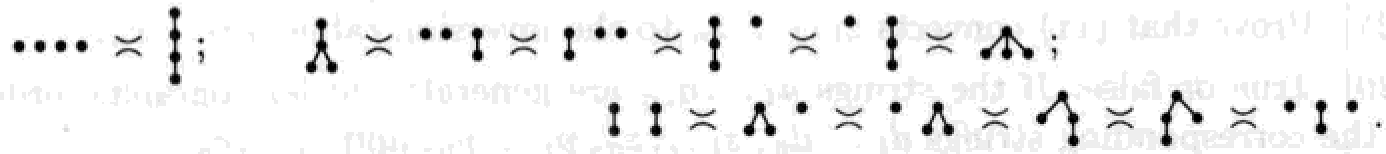
▸ **14.** [*21*] Find all labeled forests F such that $F^{RT} = F^{TR}$.

15. [*20*] Suppose B is the binary tree obtained from a forest F by linking each node to its left sibling and its rightmost child, as in exercise 2.3.2–5 and the last column of Table 2. Let F' be the forest that corresponds to B in the normal way, via left-child and right-sibling links. Prove that $F' = F^{RT}$, in the notation of exercises 11 and 12.

16. [*20*] If F and G are forests, let FG be the forest obtained by placing the trees of F to the left of the trees of G; also let $F \mid G = (G^T F^T)^T$. Give an intuitive explanation of the operator $\mid$, and prove that it is associative.

17. [*M46*] Characterize all *unlabeled* forests F such that $F^{RT} = F^{TR}$. (See exercise 14.)

18. [*30*] Two forests are said to be *cognate* if one can be obtained from the other by repeated operations of taking the conjugate and/or the transpose. The examples in exercises 11 and 12 show that all forests on 4 nodes belong to one of three cognate classes:



Study the set of all forests with 15 nodes. How many equivalence classes of cognate forests do they form? What is the largest class? What is the smallest class? What is the size of the class containing (2)?

19. [*28*] Let $F_1, F_2, \ldots, F_N$ be the sequence of unlabeled forests that correspond to the nested parentheses generated by Algorithm P, and let $G_1, G_2, \ldots, G_N$ be the sequence of unlabeled forests that correspond to the binary trees generated by Algorithm B. Prove that $G_k = F_k^{RTR}$, in the notation of exercises 11 and 12. (The forest F^{RTR} is called the *dual* of F; it is denoted by F^D in several exercises below.)

20. [*25*] Recall from Section 2.3 that the *degree* of a node in a tree is the number of children it has, and that an extended binary tree is characterized by the property that every node has degree either 0 or 2. In the extended binary tree (4), the sequence of node degrees is 2200222002220220002002202200000 in preorder; this string of 0s and 2s is identical to the sequence of parentheses in (1), except that each '(' has been replaced by 2, each ')' has been replaced by 0, and an additional 0 has been appended.

a) Prove that a sequence of nonnegative integers $b_1 b_2 \ldots b_N$ is the preorder degree sequence of a forest if and only if it satisfies the following property for $1 \le k \le N$:

$$b_1 + b_2 + \cdots + b_k + f > k \quad \text{if and only if} \quad k < N.$$

Here $f = N - b_1 - b_2 - \cdots - b_N$ is the number of trees in the forest.

b) Recall from exercise 2.3.4.5–6 that an *extended ternary tree* is characterized by the property that every node has degree 0 or 3; an extended ternary tree with n internal nodes has $2n + 1$ external nodes, hence $N = 3n + 1$ nodes altogether. Design an algorithm to generate all ternary trees with n internal nodes, by generating the associated sequences $b_1 b_2 \ldots b_N$ in lexicographic order.

▸ **21.** [*26*] (S. Zaks and D. Richards, 1979.) Continuing exercise 20, explain how to generate the preorder degree sequences of all forests that have $N = n_0 + \cdots + n_t$ nodes, with exactly n_j nodes of degree j. *Example:* When $n_0 = 4$, $n_1 = n_2 = n_3 = 1$, and $t = 3$, and the valid sequences $b_1 b_2 b_3 b_4 b_5 b_6 b_7$ are

1203000, 1230000, 1300200, 1302000, 1320000, 2013000, 2030010, 2030100, 2031000, 2103000, 2130000, 2300010, 2300100, 2301000, 2310000, 3001200, 3002010, 3002100, 3010200, 3012000, 3020010, 3020100, 3021000, 3100200, 3102000, 3120000, 3200010, 3200100, 3201000, 3210000.

▶ **22.** [*30*] (J. Korsh, 2004.) As an alternative to Algorithm B, show that binary trees can also be generated directly and efficiently in linked form if we produce them in *colex* order of the numbers $d_1 \ldots d_{n-1}$ defined in (9). (The actual values of $d_1 \ldots d_{n-1}$ should not be computed explicitly; but the links $l_1 \ldots l_n$ and $r_1 \ldots r_n$ should be manipulated in such a way that we get the binary trees corresponding successively to $d_1 d_2 \ldots d_{n-1} = 000\ldots0,\ 100\ldots0,\ 010\ldots0,\ 110\ldots0,\ 020\ldots0,\ 001\ldots0,\ \ldots,\ 000\ldots(n-1)$.)

▶ **23.** [*25*] (a) What is the last string visited by Algorithm N? (b) What is the last binary tree or forest visited by Algorithm L? *Hint:* See exercise 40 below.

24. [*22*] Using the notation of Table 3, what sequences $l_0 l_1 \ldots l_{15}$, $r_1 \ldots r_{15}$, $k_1 \ldots k_{15}$, $q_1 \ldots q_{15}$, and $u_1 \ldots u_{15}$ correspond to the binary tree (4) and the forest (2)?

▶ **25.** [*30*] (*Pruning and grafting.*) Representing binary trees as in Algorithm B, design an algorithm that visits all link tables $l_1 \ldots l_n$ and $r_1 \ldots r_n$ in such a way that, between visits, exactly one link changes from j to 0 and another from 0 to j, for some index j. (In other words, every step removes some subtree j from the binary tree and places it elsewhere, preserving preorder.)

26. [*M31*] (*The Kreweras lattice.*) Let F and F' be n-node forests with their nodes numbered 1 to n in preorder. We write $F \ltimes F'$ ("F coalesces F'") if j and k are siblings in F whenever they are siblings in F', for $1 \le j < k \le n$. Figure 39 illustrates this partial ordering in the case $n = 4$; each forest is encoded by the sequence $c_1 \ldots c_n$ of (9) and (10), which specifies the depth of each node. (With this encoding, j and k are siblings if and only if $c_j = c_k \le c_{j+1}, \ldots, c_{k-1}$.)

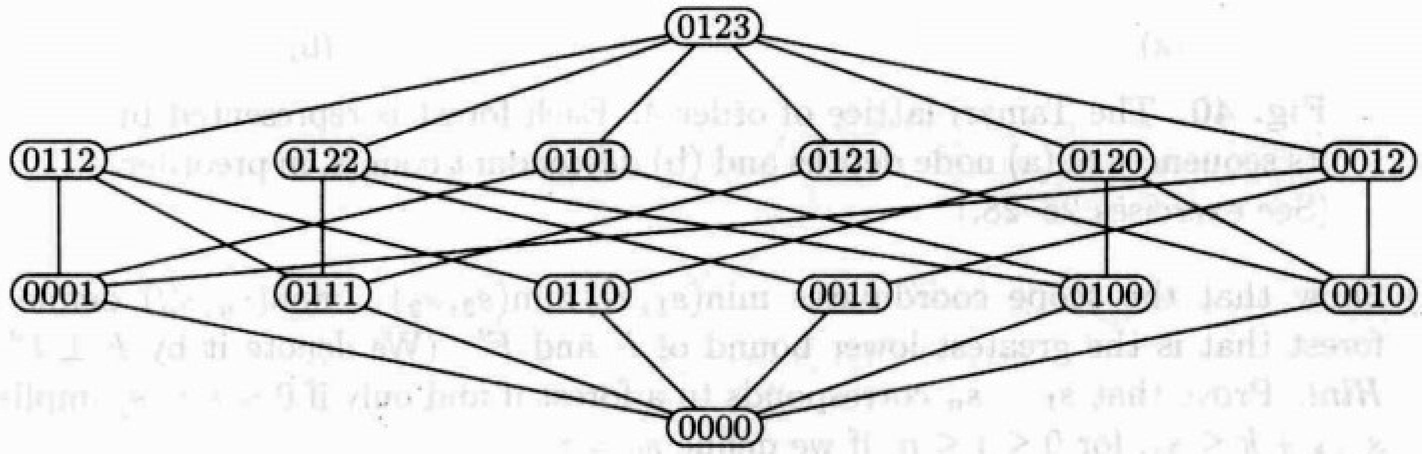


Fig. 39. The Kreweras lattice of order 4. Each forest is represented by its sequence of node depths $c_1 c_2 c_3 c_4$ in preorder. (See exercises 26–28.)

a) Let Π be a partition of $\{1, \ldots, n\}$. Show that there exists a forest F, with nodes labeled $(1, \ldots, n)$ in preorder and with

$$j \equiv k \text{ (modulo } \Pi) \iff j \text{ is a sibling of } k \text{ in } F,$$

if and only if Π satisfies the *noncrossing* property

$$i<j<k<l \text{ and } i \equiv k \text{ and } j \equiv l \text{ (modulo } \Pi) \quad \text{implies} \quad i \equiv j \equiv k \equiv l \text{ (modulo } \Pi).$$

b) Given any two n-node forests F and F', explain how to compute their least upper bound $F \veebar F'$, the element such that $F \ltimes G$ and $F' \ltimes G$ if and only if $F \veebar F' \ltimes G$.

c) When does F' cover F with respect to the relation $\ltimes$? (See exercise 7.2.1.4–55.)

d) Show that if F' covers F, it has exactly one less leaf than F.

e) How many forests cover F, when node k has e_k children for $1 \le k \le n$?

f) Using the definition of duality in exercise 19, what is the dual of the forest (2)?

g) Prove that $F \ltimes F'$ holds if and only if $F'^D \ltimes F^D$. (Because of this property, dual elements have been placed symmetrically about the center of Fig. 39.)

h) Given any two n-node forests F and F', explain how to compute their greatest lower bound $F \barwedge F'$; that is, $G \ltimes F$ and $G \ltimes F'$ if and only if $G \ltimes F \barwedge F'$.

i) Does this lattice satisfy a semimodular law analogous to exercise 7.2.1.5–12(f)?

▶ **27.** [*M33*] (*The Tamari lattice.*) Continuing exercise 26, let us write $F \dashv F'$ if the jth node in preorder has at least as many descendants in F' as it does in F, for all j. In other words, if F and F' are characterized by their scope sequences $s_1 \ldots s_n$ and $s'_1 \ldots s'_n$ as in Table 2, we have $F \dashv F'$ if and only $s_j \le s'_j$ for $1 \le j \le n$. (See Fig. 40.)

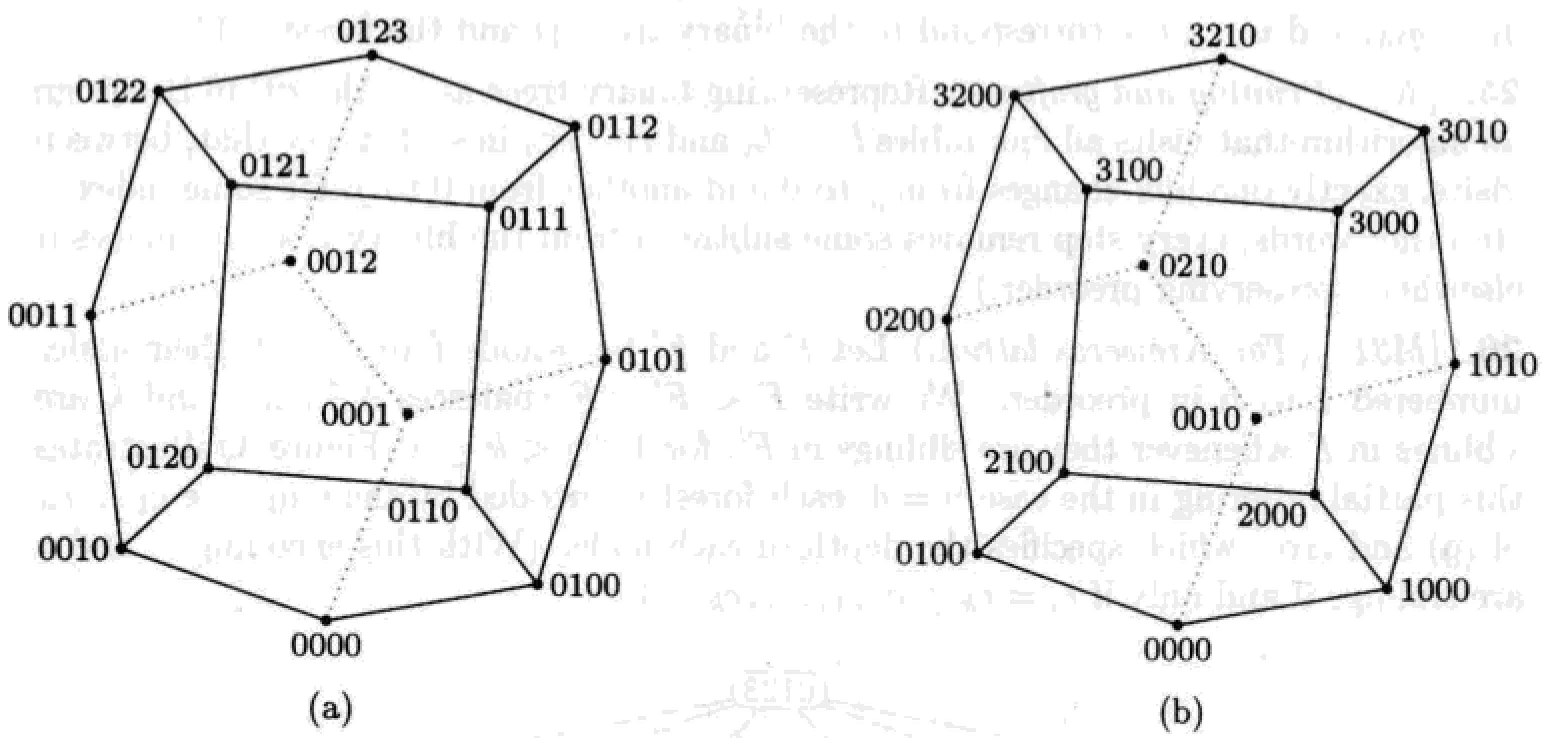


Fig. 40. The Tamari lattice of order 4. Each forest is represented by its sequences of (a) node depths and (b) descendant counts, in preorder. (See exercises 26–28.)

a) Show that the scope coordinates $\min(s_1, s'_1)\min(s_2, s'_2)\ldots\min(s_n, s'_n)$ define a forest that is the greatest lower bound of F and F'. (We denote it by $F \perp F'$.) *Hint:* Prove that $s_1 \ldots s_n$ corresponds to a forest if and only if $0 \le k \le s_j$ implies $s_{j+k} + k \le s_j$, for $0 \le j \le n$, if we define $s_0 = n$.

b) When does F' cover F in this partial ordering?

c) Prove that $F \dashv F'$ if and only if $F'^D \dashv F^D$. (Compare with exercise 26(g).)

d) Explain how to compute a least upper bound, $F \top F'$, given F and F'.

e) Prove that $F \ltimes F'$ in the Kreweras lattice implies $F \dashv F'$ in the Tamari lattice.

f) True or false: $F \barwedge F' \dashv F \perp F'$.

g) True or false: $F \veebar F' \ltimes F \top F'$.

h) What are the longest and shortest paths from the top of the Tamari lattice to the bottom, when each forest of the path covers its successor? (Such paths are called *maximal chains* in the lattice; compare with exercise 7.2.1.4–55(h).)

28. [*M26*] (*The Stanley lattice.*) Continuing exercises 26 and 27, let us define yet another partial ordering on n-node forests, saying that $F \subseteq F'$ whenever the depth coordinates $c_1 \ldots c_n$ and $c'_1 \ldots c'_n$ satisfy $c_j \le c'_j$ for $1 \le j \le n$. (See Fig. 41.)

a) Prove that this partial ordering is a lattice, by explaining how to compute the greatest lower bound $F \cap F'$ and least upper bound $F \cup F'$ of any two given forests.

b) Show that Stanley's lattice satisfies the distributive laws

$$F \cap (G \cup H) = (F \cap G) \cup (F \cap H), \qquad F \cup (G \cap H) = (F \cup G) \cap (F \cup H).$$

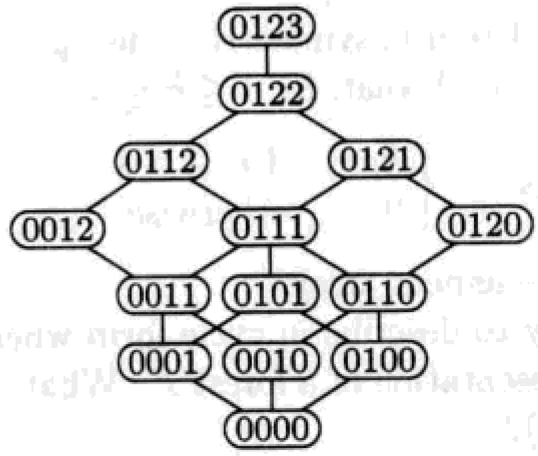


Fig. 41. The Stanley lattice of order 4. Each forest is represented by its sequence of node depths in preorder. (See exercises 26–28.)

c) When does F' cover F in this lattice?

d) True or false: $F \subseteq G$ if and only if $F^R \subseteq G^R$.

e) Prove that $F \subseteq F'$ in the Stanley lattice whenever $F \dashv F'$ in the Tamari lattice.

29. [*HM31*] The covering graph of a Tamari lattice is sometimes known as an "associahedron," because of its connection with the associative law (14), proved in exercise 27(b). The associahedron of order 4, depicted in Fig. 40, looks like it has three square faces and six faces that are regular pentagons. (Compare with Fig. 23 in exercise 7.2.1.2–60, which shows the "permutahedron" of order 4, a well-known Archimedean solid.) Why doesn't Fig. 40 show up in classical lists of uniform polyhedra?

30. [*M26*] The *footprint* of a forest is the bit string $f_1 \ldots f_n$ defined by

$$f_j = [\text{node } j \text{ in preorder is not a leaf}].$$

a) If F has footprint $f_1 \ldots f_n$, what is the footprint of F^D? (See exercise 27.)

b) How many forests have the footprint 101011011111100001010100010110000?

c) Prove that $f_j = [d_j = 0]$, for $1 \le j < n$, in the notation of (6).

d) Two elements of a lattice are called *complementary* if their greatest lower bound is the bottom element while their least upper bound is the top element. Show that F and F' are complementary in the Tamari lattice if and only if their footprints are complementary, in the sense that $f'_1 \ldots f'_{n-1} = \bar{f}_1 \ldots \bar{f}_{n-1}$.

▸ **31.** [*M28*] A binary tree with n internal nodes is called *degenerate* if it has height $n-1$.

a) How many n-node binary trees are degenerate?

b) We've seen in Tables 1, 2, and 3 that binary trees and forests can be encoded by various n-tuples of numbers. For each of the encodings $c_1 \ldots c_n$, $d_1 \ldots d_n$, $e_1 \ldots e_n$, $k_1 \ldots k_n$, $p_1 \ldots p_n$, $s_1 \ldots s_n$, $u_1 \ldots u_n$, and $z_1 \ldots z_n$, explain how to see at a glance if the corresponding binary tree is degenerate.

c) True or false: If F is degenerate, so is F^D.

d) Prove that if F and F' are degenerate, so are $F \barwedge F' = F \perp F'$ and $F \veebar F' = F \top F'$.

▸ **32.** [*M30*] Prove that if $F \dashv F'$, there is a forest F'' such that for all G we have

$$F' \perp G = F \quad \text{if and only if} \quad F \dashv G \dashv F''.$$

Consequently the *semidistributive laws* hold in the Tamari lattice:

$$F \perp G = F \perp H \quad \text{implies} \quad F \perp (G \top H) = F \perp G;$$
$$F \top G = F \top H \quad \text{implies} \quad F \top (G \perp H) = F \top G.$$

▸ **33.** [*M27*] (*Permutation representation of trees.*) Let σ be the cycle $(1\ 2\ \ldots\ n)$.

a) Given any binary tree whose nodes are numbered 1 to n in symmetric order, prove that there is a unique permutation λ of $\{1,\ldots,n\}$ such that, for $1 \le k \le n$,

$$\texttt{LLINK[}k\texttt{]} = \begin{cases} k\lambda, & \text{if } k\lambda < k; \\ 0, & \text{otherwise;} \end{cases} \qquad \texttt{RLINK[}k\texttt{]} = \begin{cases} k\sigma\lambda, & \text{if } k\sigma\lambda > k; \\ 0, & \text{otherwise.} \end{cases}$$

Thus λ neatly packs $2n$ link fields into a single n-element array.

b) Show that this permutation λ is particularly easy to describe in cycle form when the binary tree is the left-sibling/right-child representation of a forest F. What is the cycle form of $\lambda(F)$ when F is the forest in (2)?

c) Find a simple relation between $\lambda(F)$ and the dual permutation $\lambda(F^D)$.

d) Prove that, in exercise 26, F' covers F if and only if $\lambda(F') = (j\,k)\,\lambda(F)$, where j and k are siblings in F.

e) Consequently the number of maximal chains in the Kreweras lattice of order n is the number of ways to factor an n-cycle as a product of $n-1$ transpositions. Evaluate this number. *Hint:* See Eq. 1.2.6–(16).

34. [*M25*] (R. P. Stanley.) Show that the number of maximal chains in the Stanley lattice of order n is $(n(n-1)/2)!/(1^{n-1}3^{n-2}\ldots(2n-5)^2(2n-3)^1)$.

35. [*HM37*] (D. B. Tyler and D. R. Hickerson.) Explain why the denominators of the asymptotic formula (16) are all powers of 2.

▸ **36.** [*M25*] Analyze the ternary tree generation algorithm of exercise 20(b). *Hint:* There are $(2n+1)^{-1}\binom{3n}{n}$ ternary trees with n internal nodes, by exercise 2.3.4.4–11.

▸ **37.** [*M40*] Analyze the Zaks–Richards algorithm for generating all trees with a given distribution $n_0, n_1, n_2, \ldots, n_t$ of degrees (exercise 21). *Hint:* See exercise 2.3.4.4–32.

38. [*M22*] What is the total number of memory references performed by Algorithm L, as a function of n?

39. [*22*] Prove formula (23) by showing that the elements of A_{pq} in (5) correspond to Young tableaux with two rows.

40. [*M22*] (a) Prove that C_{pq} is odd if and only if $p\,\&\,(q+1) = 0$, in the sense that the binary representations of p and $q+1$ have no bits in common. (b) Therefore C_n is odd if and only if $n+1$ is a power of 2.

41. [*M21*] Show that the ballot numbers have a simple generating function $\sum C_{pq}w^pz^q$.

▸ **42.** [*M22*] How many unlabeled forests with n nodes are (a) self-conjugate? (b) self-transpose? (c) self-dual? (See exercises 11, 12, 19, and 26.)

43. [*M21*] Express C_{pq} in terms of the Catalan numbers $\langle C_0, C_1, C_2, \ldots\rangle$, aiming for a formula that is simple when $q-p$ is small. (For example, $C_{(q-2)q} = C_q - C_{q-1}$.)

▸ **44.** [*M27*] Prove that Algorithm B makes only $8\frac{2}{3} + O(n^{-1})$ references to memory per binary tree visited.

45. [*M26*] Analyze the memory references made by the algorithm in exercise 22. How does it compare to Algorithm B?

46. [*M30*] (*Generalized Catalan numbers.*) Generalize (21) by defining

$$C_{pq}(x) = C_{p(q-1)}(x) + x^{q-p}C_{(p-1)q}(x), \quad \text{if } 0 \le p \le q \ne 0; \qquad C_{00}(x) = 1;$$

and $C_{pq}(x) = 0$ if $p < 0$ or $p > q$; thus $C_{pq} = C_{pq}(1)$. Also let $C_n(x) = C_{nn}(x)$, so that $\langle C_0(x), C_1(x), \ldots\rangle = \langle 1, 1, 1+x, 1+2x+x^2+x^3, 1+3x+3x^2+3x^3+2x^4+x^5+x^6, \ldots\rangle$.

a) Show that $[x^k]\,C_{pq}(x)$ is the number of paths from (pq) to (00) in (28) that have area k, where the "area" of a path is the number of rectangular cells above it. (Thus an L-shaped path has the maximum possible area, $p(q-p)+\binom{p}{2}$.)

b) Prove that $C_n(x)=\sum_F x^{c_1+\cdots+c_n}=\sum_F x^{\text{internal path length}(F)}$, summed over all n-node forests F.

c) If $C(x,z)=\sum_{n=0}^{\infty} C_n(x)z^n$, show that $C(x,z)=1+zC(x,z)C(x,xz)$.

d) Furthermore, $C(x,z)C(x,xz)\ldots C(x,x^r z)=\sum_{p=0}^{\infty} C_{p(p+r)}(x)z^p$.

47. [*M27*] Continuing the previous exercise, generalize the identity (27).

48. [*M28*] (F. Ruskey and A. Proskurowski.) Evaluate $C_{pq}(x)$ when $x=-1$, and use this result to show that no "perfect" Gray code for nested parentheses is possible when $n\ge 5$ is odd.

49. [*17*] What is the lexicographically millionth string of 15 nested parenthesis pairs?

50. [*20*] Design the inverse of Algorithm U: Given a string $a_1\ldots a_{2n}$ of nested parentheses, determine its rank $N-1$ in lexicographic order. What is the rank of (1)?

51. [*M22*] Let $\bar z_1\bar z_2\ldots\bar z_n$ be the complement of $z_1z_2\ldots z_n$ with respect to $2n$; in other words, $\bar z_j=2n-z_j$, where z_j is defined in (8). Show that if $\bar z_1\bar z_2\ldots\bar z_n$ is the $(N+1)$st n-combination of $\{0,1,\ldots,2n-1\}$ generated by Algorithm 7.2.1.3L, then $z_1z_2\ldots z_n$ is the $(N-\kappa_nN+1)$st n-combination of $\{1,2,\ldots,2n\}$ generated by the algorithm of exercise 2. (Here κ_n denotes the nth Kruskal function, defined in 7.2.1.3–(60).)

52. [*M23*] Find the mean and variance of the quantity d_n in Table 1, when nested parentheses $a_1\ldots a_{2n}$ are chosen at random.

53. [*M28*] Let X be the distance from the root of an extended binary tree to the leftmost external node. (a) What is the expected value of X, when all binary trees with n nodes are equally likely? (b) What is the expected value of X in a random *binary search tree*, constructed by Algorithm 6.2.2T from a random permutation $K_1\ldots K_n$? (c) What is the expected value of X in a random *degenerate* binary tree, in the sense of exercise 31? (d) What is the expected value of 2^X in all three cases?

54. [*HM29*] What are the mean and variance of $c_1+\cdots+c_n$? (See exercise 46.)

55. [*HM33*] Evaluate $C'_{pq}(1)$, the total area of all the paths in exercise 46(a).

56. [*M23*] (Renzo Sprugnoli, 1990.) Prove the summation formula

$$\sum_{k=0}^{m-1} C_kC_{n-1-k} \;=\; \frac{1}{2}C_n+\frac{2m-n}{2n(n+1)}\binom{2m}{m}\binom{2n-2m}{n-m},\qquad \text{for } 0\le m\le n.$$

57. [*M28*] Express the sums $S_p(a,b)=\sum_{k\ge 0}\binom{2a}{a-k}\binom{2b}{b-k}k^p$ in closed form for $p=0$, 1, 2, 3, and use these formulas to prove (30).

58. [*HM34*] Let t_{lmn} be the number of n-node binary trees in which external node m appears at level l when the external nodes are numbered from 0 to n in symmetric order. Also let $t_{mn}=\sum_{l=1}^{n} l\,t_{lmn}$, so that t_{mn}/C_n is the average level of external node m; and let $t(w,z)$ be the super generating function

$$\sum_{m,n} t_{mn}w^m z^n=(1+w)z+(3+4w+3w^2)z^2+(9+13w+13w^2+9w^3)z^3+\cdots.$$

Prove that $t(w,z)=(C(z)-wC(wz))/(1-w)-1+zC(z)t(w,z)+wzC(wz)t(w,z)$, and deduce a simple formula for the numbers t_{mn}.

59. [*HM29*] Similarly, let T_{lmn} count all n-node binary trees in which *internal* node m appears at level l. Find a simple formula for $T_{mn} = \sum_{l=1}^{n} lT_{lmn}$.

▶ **60.** [*M26*] (*Balanced strings.*) A string α of nested parentheses is *atomic* if it has the form (α') where α' is nested; every nested string can be represented uniquely as a product of atoms $\alpha_1 \ldots \alpha_r$. A string with equal numbers of left and right parentheses is called *balanced*; every balanced string can be represented uniquely as $\beta_1 \ldots \beta_r$ where each β_j is either an atom or a *co-atom* (the reverse of an atom). The *defect* of a balanced string is half the length of its co-atoms. For example, the balanced string

(( ))) ((( )))))) (( ) ((( )) ((( )) (( )

has the factored form $\beta_1\beta_2\beta_3\beta_4\beta_5\beta_6\beta_7\beta_8 = \alpha_1\alpha_2^R\alpha_3\alpha_4^R\alpha_5^R\alpha_6\alpha_7^R\alpha_8$, with four atoms and four co-atoms; its defect is $|\alpha_2\alpha_4\alpha_5\alpha_7|/2 = 9$.

a) Prove that the defect of a balanced string is the number of indices k for which the kth right parenthesis *precedes* the kth left parenthesis.

b) If $\beta_1 \ldots \beta_r$ is balanced, we can map it into a nested string by simply reversing its co-atoms. But the following mapping is more interesting, because it produces unbiased (uniformly random) nested strings from unbiased balanced strings: Let there be s co-atoms $\beta_{i_1} = \alpha_{i_1}^R, \ldots, \beta_{i_s} = \alpha_{i_s}^R$. Replace each co-atom by (; then append the string $)\alpha'_{i_s} \ldots)\alpha'_{i_1}$, where $\alpha_j = (\alpha'_j)$. For example, the string above is mapped into $\alpha_1(\alpha_3((\alpha_6(\alpha_8)\alpha'_7)\alpha'_5)\alpha'_4)\alpha'_2$, which just happens to equal the string (1) illustrated at the beginning of this section.

Design an algorithm that applies this mapping to a given balanced string $b_1 \ldots b_{2n}$.

c) Also design an algorithm for the inverse mapping: Given a nested string $\alpha = a_1 \ldots a_{2n}$ and an integer l with $0 \le l \le n$, compute a balanced string $\beta = b_1 \ldots b_{2n}$ of defect l for which $\beta \mapsto \alpha$. What balanced string of defect 11 maps into (1)?

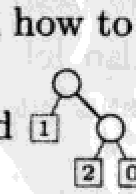
▶ **61.** [*M26*] (*Raney's Cycle Lemma.*) Let $b_1b_2 \ldots b_N$ be a string of nonnegative integers such that $f = N - b_1 - b_2 - \cdots - b_N > 0$.

a) Prove that exactly f of the cyclic shifts $b_{j+1} \ldots b_N b_1 \ldots b_j$ for $1 \le j \le N$ satisfy the preorder degree sequence property in exercise 20.

b) Design an efficient algorithm to determine all such j, given $b_1b_2 \ldots b_N$.

c) Explain how to generate a random forest that has $N = n_0 + \cdots + n_t$ nodes, with exactly n_j nodes of degree j. (For example, we obtain random n-node t-ary trees as a special case of this general procedure when $N = tn + 1$, $n_0 = (t-1)n + 1$, $n_1 = \cdots = n_{t-1} = 0$, and $n_t = n$.)

62. [*22*] A binary tree can also be represented by bit strings $(l_1 \ldots l_n, r_1 \ldots r_n)$, where l_j and r_j tell whether the left and right subtrees of node j in preorder are nonempty. (See Theorem 2.3.1A.) Prove that if $l_1 \ldots l_n$ and $r_1 \ldots r_n$ are arbitrary bit strings with $l_1 + \cdots + l_n + r_1 + \cdots + r_n = n - 1$, exactly one cyclic shift $(l_{j+1} \ldots l_n l_1 \ldots l_j, r_{j+1} \ldots r_n r_1 \ldots r_j)$ yields a valid binary tree representation, and explain how to find it.

63. [*16*] If the first two iterations of Rémy's algorithm have produced 

, what decorated binary trees are possible after the next iteration?

64. [*20*] What sequence of X values in Algorithm R corresponds to the decorated trees of (34), and what are the final values of $L_0L_1 \ldots L_{12}$?

65. [*38*] Generalize Rémy's algorithm (Algorithm R) to t-ary trees.

66. [*21*] A *Schröder tree* is a binary tree in which every nonnull right link is colored either white or black. The number S_n of n-node Schröder trees is

$n =$	0	1	2	3	4	5	6	7	8	9	10	11	12
$S_n =$	1	1	3	11	45	197	903	4279	20793	103049	518859	2646723	13648869

for small n. For example, $S_3 = 11$ because the possibilities are



(White links are "hollow"; external nodes have also been attached.)

a) Find a simple correspondence between Schröder trees with n internal nodes and ordinary trees with $n+1$ leaves and no nodes of degree one.

b) Devise a Gray code for Schröder trees.

67. [*M22*] What is the generating function $S(z) = \sum_n S_n z^n$ for Schröder numbers?

68. [*10*] What is the Christmas tree pattern of order 0?

69. [*20*] Are the Christmas tree patterns of orders 6 and 7 visible in Table 4, possibly in slight disguise?

▸ **70.** [*20*] Find a simple rule that defines, for every bit string σ, another bit string σ' called its *mate*, with the following properties: (i) $\sigma'' = \sigma$; (ii) $|\sigma'| = |\sigma|$; (iii) either $\sigma \subseteq \sigma'$ or $\sigma' \subseteq \sigma$; (iv) $\nu(\sigma) + \nu(\sigma') = |\sigma|$.

71. [*M21*] Let M_{tn} be the size of the largest possible set S of n-bit strings with the property that, if σ and τ are members of S with $\sigma \subseteq \tau$, then $\nu(\tau) < \nu(\sigma) + t$. (Thus, for example, $M_{1n} = M_n$ by Sperner's theorem.) Find a formula for M_{tn}.

▸ **72.** [*M28*] If you start with a single row $\sigma_1\ \sigma_2\ \ldots\ \sigma_s$ of length s and apply the growth rule (36) repeatedly n times, how many rows do you obtain?

73. [*15*] In the Christmas tree pattern of order 30, what are the first and last elements of the row that contains the bit string 011001001000011111101101011100?

74. [*M26*] Continuing the previous exercise, how many rows precede that row?

▸ **75.** [*HM23*] Let $(r_1^{(n)}, r_2^{(n)}, \ldots, r_{n-1}^{(n)})$ be the row numbers in which the Christmas tree pattern of order n has $n-1$ entries; for example, Table 4 tells us that $(r_1^{(8)}, \ldots, r_7^{(8)}) = (20, 40, 54, 62, 66, 68, 69)$. Find formulas for $r_{j+1}^{(n)} - r_j^{(n)}$ and for $\lim_{n\to\infty} r_j^{(n)}/M_n$.

76. [*HM46*] Study the limiting shape of the Christmas tree patterns as $n \to \infty$. Does it, for example, have a fractal dimension under some appropriate scaling?

77. [*21*] Design an algorithm to generate the sequence of rightmost elements $a_1 \ldots a_n$ in the rows of the Christmas tree pattern, given n. *Hint:* These bit strings are characterized by the property that $a_1 + \cdots + a_k \geq k/2$ for $0 \leq k \leq n$.

78. [*20*] True or false: If $\sigma_1\ \ldots\ \sigma_s$ is a row of the Christmas tree pattern, so is $\bar{\sigma}_s^R\ \ldots\ \bar{\sigma}_1^R$ (the reverse sequence of reverse complements).

79. [*M26*] The number of permutations $p_1 \ldots p_n$ that have exactly one "descent" where $p_k > p_{k+1}$ is the Eulerian number $\left\langle {n \atop 1} \right\rangle = 2^n - n - 1$, according to Eq. 5.1.3–(12). The number of entries in the Christmas tree pattern, above the bottom row, is the same.

a) Find a combinatorial explanation of this coincidence, by giving a one-to-one correspondence between one-descent permutations and unsorted bit strings.

b) Show that two unsorted bit strings belong to the same row of the Christmas tree pattern if and only if they correspond to permutations that define the same P tableau under the Robinson–Schensted correspondence (Theorem 5.1.4A).

80. [*30*] Say that two bit strings are *concordant* if we can obtain one from the other via the transformations 010 $\leftrightarrow$ 100 or 101 $\leftrightarrow$ 110 on substrings. For example, the strings

$$\begin{array}{ccccccccc} 011100 & \leftrightarrow & 011010 & \leftrightarrow & 010110 & \leftrightarrow & 010101 & \leftrightarrow & 011001 \\ & & & & \updownarrow & & \updownarrow & & \\ & & & & 100110 & \leftrightarrow & 100101 & \leftrightarrow & 101001 \leftrightarrow 110001 \end{array}$$

are mutually concordant, but no other string is concordant with any of them.

Prove that strings are concordant if and only if they belong to the same column of the Christmas tree pattern and to rows of the same length in that pattern.

81. [*M30*] A *biclutter* of order (n, n') is a family S of bit string pairs (σ, σ'), where $|\sigma| = n$ and $|\sigma'| = n'$, with the property that distinct members (σ, σ') and (τ, τ') of S are allowed to satisfy $\sigma \subseteq \tau$ and $\sigma' \subseteq \tau'$ only if $\sigma \neq \tau$ and $\sigma' \neq \tau'$.

Use Christmas tree patterns to prove that S contains at most $M_{n+n'}$ string pairs.

▸ **82.** [*M26*] Let $E(f)$ be the number of times Algorithm H evaluates the function f.

a) Show that $M_n \leq E(f) \leq M_{n+1}$, with equality when f is constant.

b) Among all f such that $E(f) = M_n$, which one minimizes $\sum_\sigma f(\sigma)$?

c) Among all f such that $E(f) = M_{n+1}$, which one maximizes $\sum_\sigma f(\sigma)$?

83. [*M20*] (G. Hansel.) Show that there are at most 3^{M_n} monotone Boolean functions $f(x_1, \ldots, x_n)$ of n Boolean variables.

▸ **84.** [*HM27*] (D. Kleitman.) Let A be an $m \times n$ matrix of real numbers in which every column v has length $\|v\| \geq 1$, and let b be an m-dimensional column vector. Prove that at most M_n column vectors $x = (a_1, \ldots, a_n)^T$, with components $a_j = 0$ or 1, satisfy $\|Ax - b\| < \frac{1}{2}$. *Hint:* Use a construction analogous to the Christmas tree pattern.

85. [*HM35*] (Philippe Golle.) Let V be any vector space contained in the set of all real n-dimensional vectors, but containing none of the unit vectors $(1, 0, \ldots, 0)$, $(0, 1, 0, \ldots, 0)$, …, $(0, \ldots, 0, 1)$. Prove that V contains at most M_n vectors whose components are all 0 or 1; furthermore the upper bound M_n is achievable.

86. [*15*] If (2) is regarded as an *oriented forest* instead of an ordered forest, what canonical forest corresponds to it? Specify that forest both by its level codes $c_1 \ldots c_{15}$ and its parent pointers $p_1 \ldots p_{15}$.

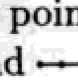
87. [*M20*] Let F be an ordered forest in which the kth node in preorder appears on level c_k and has parent p_k, where $p_k = 0$ if that node is a root.

a) How many forests satisfy the condition $c_k = p_k$ for $1 \leq k \leq n$?

b) Suppose F and F' have level codes $c_1 \ldots c_n$ and $c'_1 \ldots c'_n$, respectively, as well as parent links $p_1 \ldots p_n$ and $p'_1 \ldots p'_n$. Prove that, lexicographically, $c_1 \ldots c_n \leq c'_1 \ldots c'_n$ if and only if $p_1 \ldots p_n \leq p'_1 \ldots p'_n$.

88. [*M20*] Analyze Algorithm O: How often is step O4 performed? What is the total number of times p_k is changed in step O5?

89. [*M46*] How often does step O5 set $p_k \leftarrow p_j$?

▸ **90.** [*M27*] If $p_1 \ldots p_n$ is a canonical sequence of parent pointers for an oriented forest, the graph with vertices $\{0, 1, \ldots, n\}$ and edges $\{k — p_k \mid 1 \leq k \leq n\}$ is a *free tree*, namely a connected graph with no cycles. (See Theorem 2.3.4.1A.) Conversely, every free tree corresponds to at least one oriented forest in this way. But the parent pointers 011 and 000 both yield the same free tree ; similarly, 012 and 010 both yield .

The purpose of this exercise is to restrict the sequences $p_1 \ldots p_n$ further so that each free tree is obtained exactly once. We proved in 2.3.4.4–(9) that the number of structurally different free trees on $n+1$ vertices has a fairly simple generating function, by showing that a free tree always has at least one *centroid*.

a) Show that a canonical n-node forest corresponds to a free tree with a single centroid if and only if no tree in the forest has more than $\lfloor n/2 \rfloor$ nodes.

b) Modify Algorithm O so that it generates all sequences $p_1 \ldots p_n$ that satisfy (a).

c) Explain how to find all $p_1 \ldots p_n$ for free trees that have *two* centroids.

91. [*M37*] (Nijenhuis and Wilf.) Show that a random oriented tree can be generated with a procedure analogous to the random partition algorithm of exercise 7.2.1.4–47.

92. [*15*] Are the first and last spanning trees visited by Algorithm S adjacent, in the sense that they have $n-2$ edges in common?

93. [*20*] When Algorithm S terminates, has it restored the graph to its original state?

94. [*22*] Algorithm S needs to "prime the pump" by finding an initial spanning tree in step S1. Explain how to do that task.

95. [*26*] Complete Algorithm S by implementing the bridge test in step S8.

▶ **96.** [*28*] Analyze the approximate running time of Algorithm S when the given graph is simply (a) a path P_n of length $n-1$; (b) a cycle C_n of length n.

97. [*15*] Is (48) a series-parallel graph?

98. [*16*] What series-parallel graph corresponds to (53) if A is taken to be *serial*?

▶ **99.** [*30*] Consider a series-parallel graph represented by a tree as in (53), together with node values that satisfy (55). These values define a spanning tree or a near tree, according as v_p is 1 or 0 at the root p. Show that the following method will generate all of the other configs of the root:

i) Begin with all uneasy nodes active, other nodes passive.

ii) Select the rightmost active node, p, in preorder; but terminate if all nodes are passive.

iii) Change $d_p \leftarrow r_{d_p}$, update all values in the tree, and visit the new config.

iv) Activate all uneasy nodes to the right of p.

v) If d_p has run through all children of p since p last became active, make node p passive. Return to (ii).

Also explain how to perform these steps efficiently. *Hints:* To implement step (v), introduce a pointer z_p; make node p passive when d_p becomes equal to z_p, and at such times also reset z_p to the previous value of d_p. To implement steps (ii) and (iv), use focus pointers f_p analogous to those in Algorithms 7.2.1.1L and 7.2.1.1K.

100. [*40*] Implement the text's "Algorithm S′" for revolving-door generation of all spanning trees, by combining Algorithm S with the ideas of exercise 99.

101. [*46*] Is there a simple revolving-door way to list all n^{n-2} spanning trees of the complete graph K_n? (The order produced by Algorithm S is quite complicated.)

102. [*46*] An *oriented spanning tree* of a directed graph D on n vertices, also known as a "spanning arborescence," is an oriented subtree of D containing $n-1$ arcs. The matrix tree theorem (exercise 2.3.4.2–19) tells us that the oriented subtrees having a given root can readily be counted by evaluating an $(n-1) \times (n-1)$ determinant.

Can those oriented subtrees be listed in a revolving-door order, always removing one arc and replacing it with another?

▶ **103.** [*HM39*] (*Sandpiles.*) Consider any digraph D on vertices $V_0, V_1, \ldots, V_n$ with e_{ij} arcs from V_i to V_j, where $e_{ii} = 0$. Assume that D has at least one oriented spanning tree rooted at V_0; this assumption means that, if we number the vertices appropriately, we have $e_{i0} + \cdots + e_{i(i-1)} > 0$ for $1 \le i \le n$. Let $d_i = e_{i0} + \cdots + e_{in}$ be the total out-degree of V_i. Put x_i grains of sand on vertex V_i for $0 \le i \le n$, and play the following game: If $x_i \ge d_i$ for any $i \ge 1$, decrease x_i by d_i and set $x_j \leftarrow x_j + e_{ij}$ for all $j \ne i$. (In other words, pass one grain of sand from V_i through each of its outgoing arcs, whenever possible, except when $i = 0$. This operation is called "toppling" V_i, and a sequence of topplings is called an "avalanche." Vertex V_0 is special; instead of toppling, it collects particles of sand that essentially leave the system.) Continue until $x_i < d_i$ for $1 \le i \le n$. Such a state $x = (x_1, \ldots, x_n)$ is called *stable*.

a) Prove that every avalanche terminates in a stable state after a finite number of topplings. Furthermore, the final state depends only on the initial state, not on the order in which toppling is performed.

b) Let $\sigma(x)$ be the stable state that results from initial state x. A stable state is called *recurrent* if it is $\sigma(x)$ for some x with $x_i \ge d_i$ for $1 \le i \le n$. (Recurrent states correspond to sandpiles that have evolved over a long period of time, after new grains of sand are repeatedly introduced at random.) Find the recurrent states in the special case when $n = 4$ and when the only arcs of D are

$$V_1 \to V_0,\ V_1 \to V_2,\ V_2 \to V_0,\ V_2 \to V_1,\ V_3 \to V_0,\ V_3 \to V_4,\ V_4 \to V_0,\ V_4 \to V_3.$$

c) Let $d = (d_1, \ldots, d_n)$. Prove that x is recurrent if and only if $x = \sigma(x + t)$, where t is the vector $d - \sigma(d)$.

d) Let a_i be the vector $(-e_{i1}, \ldots, -e_{i(i-1)}, d_i, -e_{i(i+1)}, \ldots, -e_{in})$, for $1 \le i \le n$; thus, toppling V_i corresponds to changing the state vector $x = (x_1, \ldots, x_n)$ to $x - a_i$. Say that two states x and x' are *congruent*, written $x \equiv x'$, if $x - x' = m_1 a_1 + \cdots + m_n a_n$ for some integers $m_1, \ldots, m_n$. Prove that there are exactly as many equivalence classes of congruent states as there are oriented spanning trees in D, rooted at V_0. *Hint:* See the matrix tree theorem, exercise 2.3.4.2–19.

e) If $x \equiv x'$ and if both x and x' are recurrent, prove that $x = x'$.

f) Prove that every congruence class contains a unique recurrent state.

g) If D is *balanced*, in the sense that the in-degree of each vertex equals its out-degree, prove that x is recurrent if and only if $x = \sigma(x + a)$, where $a = (e_{01}, \ldots, e_{0n})$.

h) Illustrate these concepts when D is a "wheel" with n spokes: Let there be $3n$ arcs, $V_j \to V_0$ and $V_j \leftrightarrow V_{j+1}$ for $1 \le j \le n$, regarding V_{n+1} as identical to V_1. Find a one-to-one correspondence between the oriented spanning trees of this digraph and the recurrent states of its sandpiles.

i) Similarly, analyze the recurrent sandpiles when D is the *complete* graph on $n+1$ vertices, namely when $e_{ij} = [i \ne j]$ for $0 \le i, j \le n$. *Hint:* See exercise 6.4–31.

▶ **104.** [*HM21*] If G is a graph on n vertices $\{V_1, \ldots, V_n\}$, with e_{ij} edges between V_i and V_j, let $C(G)$ be the matrix with entries $c_{ij} = -e_{ij} + \delta_{ij} d_i$, where $d_i = e_{i1} + \cdots + e_{in}$ is the degree of V_i. Let us say that the *aspects* of G are the eigenvalues of $C(G)$, namely the roots $\alpha_0, \ldots, \alpha_{n-1}$ of the equation $\det(\alpha I - C(G)) = 0$. Since $C(G)$ is a symmetric matrix, its eigenvalues are real numbers, and we can assume that $\alpha_0 \le \alpha_1 \le \cdots \le \alpha_{n-1}$.

a) Prove that $\alpha_0 = 0$.

b) Prove that G has exactly $c(G) = \alpha_1 \ldots \alpha_{n-1}/n$ spanning trees.

c) What are the aspects of the complete graph K_n?

105. [*HM37*] Continuing exercise 104, we wish to prove that there is often an easy way to determine the aspects of G when G has been constructed from other graphs whose aspects are known. Suppose G' has aspects $\alpha'_0, \ldots, \alpha'_{n'-1}$ and G'' has aspects $\alpha''_0, \ldots, \alpha''_{n''-1}$; what are the aspects of G in the following cases?

a) $G = \overline{G'}$ is the complement of G'. (Assume that $e'_{ij} \le [i \ne j]$ in this case.)

b) $G = G' + G''$ is the sum (juxtaposition) of G' and G''.

c) $G = G' \mathbin{\bar{+}} G''$ is the cosum (join) of G' and G''.

d) $G = G' \times G''$ is the Cartesian product of G' and G''.

e) $G = L(G')$ is the line graph of G', when G' is a regular graph of degree d' (namely when all vertices of G' have exactly d' neighbors, and there are no self-loops).

f) $G = G' \diamond G''$ is the direct product (conjunction) of G' and G'', when G' is regular of degree d' and G'' is regular of degree d''.

g) $G = G' \otimes G''$ is the strong product of regular graphs G' and G''.

▶ **106.** [*HM37*] Find the total number of spanning trees in (a) an $m \times n$ grid $P_m \times P_n$; (b) an $m \times n$ cylinder $P_m \times C_n$; (c) an $m \times n$ torus $C_m \times C_n$. Why do these numbers tend to have only small prime factors? *Hint:* Show that the aspects of P_n and C_n can be expressed in terms of the numbers $\sigma_{kn} = 4\sin^2 \frac{k\pi}{2n}$.

107. [*M24*] Determine the aspects of all connected graphs that have $n \le 5$ vertices and no self-loops or parallel edges.

108. [*HM40*] Extend the results of exercises 104–106 to directed graphs.

109. [*M46*] Find a combinatorial explanation for the fact that (57) is the number of spanning trees in the n-cube.

▶ **110.** [*M27*] Prove that if G is any connected multigraph without self-loops, it has

$$c(G) > \sqrt{(d_1 - 1) \ldots (d_n - 1)}$$

spanning trees, where d_j is the degree of vertex j.

111. [*05*] List the nodes of the tree (58) in postpreorder.

112. [*15*] If node p of a forest precedes node q in prepostorder and follows it in postpreorder, what can you say about p and q?

▶ **113.** [*20*] How do prepostorder and postpreorder of a forest F relate to prepostorder and postpreorder of the conjugate forest F^R? (See exercise 13.)

114. [*15*] If we want to traverse an entire forest in prepostorder using Algorithm Q, how should we begin the process?

115. [*20*] Analyze Algorithm Q: How often is each step performed, during the complete traversal of a forest?

▶ **116.** [*28*] If the nodes of a forest F are labeled 1 to n in prepostorder, say that node k is *lucky* if it is adjacent to node $k+1$ in F, *unlucky* if it is three steps away, and *ordinary* otherwise, for $1 \le k \le n$; in this definition, node $n+1$ is an imaginary super-root considered to be the parent of each root.

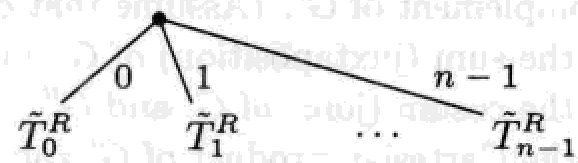
a) Prove that lucky nodes occur only on even-numbered levels; unlucky nodes occur only on odd-numbered levels.

b) Show that the number of lucky nodes is exactly one greater than the number of unlucky nodes, unless $n = 0$.

117. [*21*] How many n-node forests contain no unlucky nodes?

118. [*M28*] How many lucky nodes are present in (a) the complete t-ary tree with $(t^k-1)/(t-1)$ internal nodes? (b) the Fibonacci tree of order k, with $F_{k+1}-1$ internal nodes? (See 2.3.4.5–(6) and Fig. 8 in Section 6.2.1.)

119. [*21*] The *twisted binomial tree* $\tilde{T}_n$ of order n is defined recursively by the rules

$\tilde{T}_0 = \bullet$, $\tilde{T}_n =$  for $n > 0$.

(Compare with 7.2.1.3–(21); we reverse the order of children on alternate levels.) Show that prepostorder traversal of $\tilde{T}_n$ has a simple connection with Gray binary code.

120. [*22*] True or false: The square of a graph is Hamiltonian if the graph is connected and has no bridges.

121. [*M32*] (F. Neuman, 1964.) The *derivative* of a graph G is the graph $G^{(\prime)}$ obtained by removing all vertices of degree 1 and the edges touching them. Prove that, when T is a free tree, its square T^2 contains a Hamiltonian path if and only if its derivative has no vertex of degree greater than 4 and the following two additional conditions hold:

i) All vertices of degree 3 or 4 in $T^{(\prime)}$ lie on a single path.

ii) Between any two vertices of degree 4 in $T^{(\prime)}$, there is at least one vertex that has degree 2 in T.

▸ **122.** [*31*] (*Dudeney's Digital Century puzzle.*) There are many curious ways to obtain the number 100 by inserting arithmetical operators and possibly also parentheses into the sequence 123456789. For example,

$$100 = 1+2\times 3+4\times 5-6+7+8\times 9 = (1+2-3-4)\times(5-6-7-8-9)$$
$$= ((1/((2+3)/4-5+6))\times 7+8)\times 9\,.$$

a) How many such representations of 100 are possible? To make this question precise, in view of the associative law and other algebraic properties, assume that expressions are written in canonical form according to the following syntax:

$\langle\text{expression}\rangle \to \langle\text{number}\rangle \mid \langle\text{sum}\rangle \mid \langle\text{product}\rangle \mid \langle\text{quotient}\rangle$
$\langle\text{sum}\rangle \to \langle\text{term}\rangle + \langle\text{term}\rangle \mid \langle\text{term}\rangle - \langle\text{term}\rangle \mid \langle\text{sum}\rangle + \langle\text{term}\rangle \mid \langle\text{sum}\rangle - \langle\text{term}\rangle$
$\langle\text{term}\rangle \to \langle\text{number}\rangle \mid \langle\text{product}\rangle \mid \langle\text{quotient}\rangle$
$\langle\text{product}\rangle \to \langle\text{factor}\rangle \times \langle\text{factor}\rangle \mid \langle\text{product}\rangle \times \langle\text{factor}\rangle \mid (\langle\text{quotient}\rangle) \times \langle\text{factor}\rangle$
$\langle\text{quotient}\rangle \to \langle\text{factor}\rangle / \langle\text{factor}\rangle \mid \langle\text{product}\rangle / \langle\text{factor}\rangle \mid (\langle\text{quotient}\rangle) / \langle\text{factor}\rangle$
$\langle\text{factor}\rangle \to \langle\text{number}\rangle \mid (\langle\text{sum}\rangle)$
$\langle\text{number}\rangle \to \langle\text{digit}\rangle$

The digits used must be 1 through 9, in that order.

b) Extend problem (a) by allowing multidigit numbers, with the syntax

$$\langle\text{number}\rangle \to \langle\text{digit}\rangle \mid \langle\text{number}\rangle\langle\text{digit}\rangle$$

For example, $100 = (1/(2-3+4))\times 567-89$. What is the shortest such representation? What is the longest?

c) Extend problem (b) by also allowing decimal points:

$$\langle\text{number}\rangle \to \langle\text{digit string}\rangle \mid .\langle\text{digit string}\rangle$$
$$\langle\text{digit string}\rangle \to \langle\text{digit}\rangle \mid \langle\text{digit string}\rangle\langle\text{digit}\rangle$$

For example, $100 = (.1-2-34\times .5)/(.6-.789)$, amazingly enough.

123. [*21*] Continuing the previous exercise, what are the smallest positive integers that *cannot* be represented using conventions (a), (b), (c)?

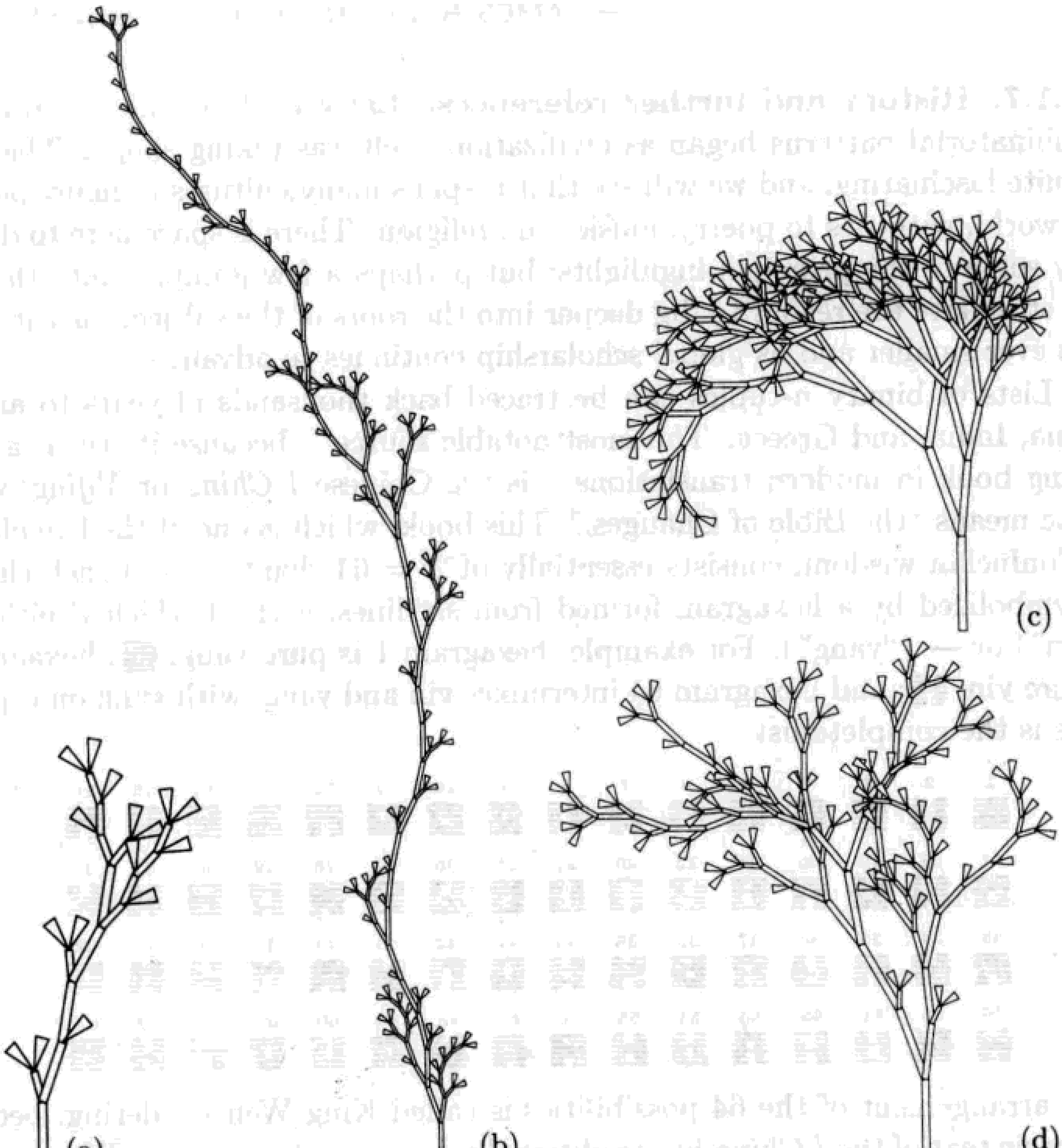


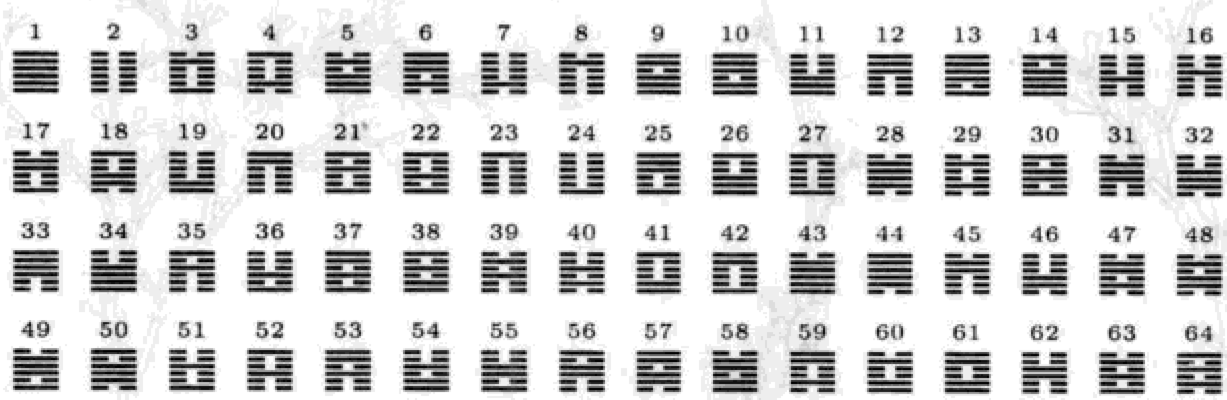
Fig. 42. "Organic" illustrations of binary trees.

▸ **124.** [*40*] Experiment with methods for drawing extended binary trees that are inspired by simple models from nature. For example, we can assign a value $v(x)$ to each node x, called its *Horton–Strahler number*, as follows: Each external (leaf) node has $v(x) = 0$; an internal node with children (l, r) has $v(x) = \max(v(l), v(r)) + [v(l) = v(r)]$. The edge from internal node x to its parent can be drawn as a rectangle with height $h(v(x))$ and width $w(v(x))$, and the edge rectangles with children (l, r) can be offset by angles $\theta(v(l(x)), v(r(x)))$, $-\theta(v(r(x)), v(l(x)))$, for certain functions h, w, and θ. The examples in Fig. 42 show typical results when we choose $w(k) = 3 + k$, $h(k) = 18k$, $\theta(k, k) = 30°$, $\theta(j, k) = ((k + 1)/j) \times 20°$ for $0 \le k < j$, and $\theta(j, k) = ((k - j)/k) \times 30°$ for $0 \le j < k$; the roots appear at the bottom. Part (a) of Fig. 42 is the binary tree (4): part (b) is a random 100-node tree generated by Algorithm R; part (c) is the Fibonacci tree of order 11, which has 143 nodes; and part (d) is a random 100-node binary search tree. (The trees in parts (b), (c), and (d) clearly belong to different species.)

> *[This subject] has a relation*
> *to almost every species of useful knowledge*
> *that the mind of man can be employed upon.*
> — JAMES BERNOULLI, *Ars Conjectandi* (1713)

7.2.1.7. History and further references. Early work on the generation of combinatorial patterns began as civilization itself was taking shape. The story is quite fascinating, and we will see that it spans many cultures in many parts of the world, with ties to poetry, music, and religion. There is space here to discuss only some of the principal highlights; but perhaps a few glimpses into the past will stimulate the reader to dig deeper into the roots of the subject, as the world gets ever smaller and as global scholarship continues to advance.

Lists of binary n-tuples can be traced back thousands of years to ancient China, India, and Greece. The most notable source — because it still is a best-selling book in modern translations — is the Chinese *I Ching* or *Yijing*, whose name means "the Bible of Changes." This book, which is one of the five classics of Confucian wisdom, consists essentially of $2^6 = 64$ chapters; and each chapter is symbolized by a hexagram formed from six lines, each of which is either -- ("yin") or — ("yang"). For example, hexagram 1 is pure yang, ䷀; hexagram 2 is pure yin, ䷁; and hexagram 64 intermixes yin and yang, with yang on top: ䷿. Here is the complete list:



(1)

This arrangement of the 64 possibilities is called King Wen's ordering, because the basic text of the *I Ching* has traditionally been ascribed to King Wen (c. 1100 B.C.), the legendary progenitor of the Chou dynasty. Ancient texts are, however, notoriously difficult to date reliably, and modern historians have found no solid evidence that anyone actually compiled such a list of hexagrams before the third century B.C.

Notice that the hexagrams of (1) occur in pairs: Those with odd numbers are immediately followed by their top-to-bottom reflections, except when reflection would make no change; and the eight symmetrical diagrams are paired with their complements ($1 = \overline{2}$, $27 = \overline{28}$, $29 = \overline{30}$, $61 = \overline{62}$). Hexagrams that are composed from two trigrams that represent the four basic elements heaven (☰), earth (☷), fire (☲), and water (☵) have also been placed judiciously. Otherwise the arrangement appears to be essentially random, as if a person untrained in mathematics kept listing different possibilities until being unable to come up with any more. A few intriguing patterns do exist between the pairs, but no more than are present by coincidence in the digits of π (see 3.3–(1)).

Yin and yang represent complementary aspects of the elementary forces of nature, always in tension, always changing. The *I Ching* is somewhat analogous to a thesaurus in which the hexagrams serve as an index to accumulated wisdom about fundamental concepts like giving (䷀), receiving (䷁), modesty (䷎), joy (䷹), fellowship (䷌), withdrawal (䷠), peace (䷊), conflict (䷅), organization (䷆), corruption (䷑), immaturity (䷃), elegance (䷕), etc. One can choose a pair of hexagrams at random, obtaining the second from the first by, say, independently changing each yin to yang (or vice versa) with probability 1/4; this technique yields 4096 ways to ponder existential mysteries, as well as a Markov process by which change itself might perhaps give meaning to life.

A strictly logical way to arrange the hexagrams was eventually introduced about A.D. 1060 by Shao Yung. His ordering, which proceeded lexicographically from ䷁ to ䷗ to ䷆ to ䷒ to ䷎ to $\cdots$ to ䷪ to ䷀ (reading each hexagram from bottom to top), was much more user-friendly than the King Wen order, because a random pattern could now be found quickly. When G. W. Leibniz learned about this sequence of hexagrams in 1702, he jumped to the erroneous conclusion that Chinese mathematicians had once been familiar with binary arithmetic. [See Frank Swetz, *Mathematics Magazine* **76** (2003), 276–291. Further details about the *I Ching* can be found, for example, in Joseph Needham's *Science and Civilisation in China* **2** (Cambridge University Press, 1956), 304–345; R. J. Lynn, *The Classic of Changes* (New York: Columbia University Press, 1994).]

Another ancient Chinese philosopher, Yang Hsiung, proposed a system based on 81 ternary tetragrams instead of 64 binary hexagrams. His *Canon of Supreme Mystery*, written c. 2 B.C., has recently been translated into English by Michael Nylan (Albany, New York: 1993). Yang described a complete, hierarchical ternary tree structure in which there are 3 regions, with 3 provinces in each region, 3 departments in each province, 3 families in each department, and 9 short poems called "appraisals" for each family, hence 729 appraisals in all — making almost exactly 2 appraisals for every day in the year. His tetragrams were arranged in strict lexicographic order when read top-to-bottom: 𝌆, 𝌇, 𝌈, 𝌉, 𝌊, 𝌋, 𝌌, …, 𝍖. In fact, as explained on page 28 of Nylan's book, Yang presented a simple way to compute the rank of each tetragram, as if using a radix-3 number system. Thus he would not have been surprised or impressed by Shao Yung's systematic ordering of binary hexagrams, although Shao lived more than 1000 years later.

Indian prosody. Binary n-tuples were studied in a completely different context by pundits in ancient India, who investigated the poetic meters of sacred Vedic chants. Syllables in Sanskrit are either short (।) or long (ऽ), and the study of syllable patterns is called "prosody." Modern writers use the symbols ⏑ and — instead of । and ऽ. A typical Vedic verse consists of four lines with n syllables per line, for some $n \geq 8$; prosodists therefore sought a way to classify all 2^n possibilities. The classic work *Chandaḥśāstra* by Piṅgala, written before A.D. 400 and probably much earlier (the exact date is quite uncertain), described procedures by which one could readily find the index k of any given pattern of ⏑s and —s, as well as to find the kth pattern, given k. In other words, Piṅgala explained how to *rank* any given pattern as well as to *unrank* any given index;

thus he went beyond the work of Yang Hsiung, who had considered ranking but not unranking. Piṅgala's methods were also related to exponentiation, as we have noted earlier in connection with Algorithm 4.6.3A.

The next important step was taken by a prosodist named Kedāra in his work *Vṛttaratnākara*, thought to have been written in the 8th century. Kedāra gave a step-by-step procedure for listing all the n-tuples from ———…— to ⌣——…— to —⌣—…— to ⌣⌣—…— to ——⌣…— to ⌣—⌣…— to ⋯ to ⌣⌣⌣…⌣, essentially Algorithm 7.2.1.1M in the case of radix 2. His method may well have been the first-ever explicit algorithm for combinatorial sequence generation. [See B. van Nooten, *J. Indian Philos.* **21** (1993), 31–50.]

Poetic meters can also be regarded as rhythms, with one beat for each ⌣ and two beats for each —. An n-syllable pattern can involve between n and $2n$ beats, but musical rhythms suitable for marching or dancing generally are based on a fixed number of beats. Therefore it was natural to consider the set of all sequences of ⌣s and —s that have exactly m beats, for fixed m. Such patterns are now called Morse code sequences of length m, and we know from exercise 4.5.3–32 that there are exactly F_{m+1} of them. For example, the 21 sequences when $m = 7$ are

⌣———, —⌣——, ⌣⌣⌣——, ——⌣—, ⌣⌣—⌣—,
⌣—⌣⌣—, —⌣⌣⌣—, ⌣⌣⌣⌣⌣—, ———⌣,
⌣⌣——⌣, ⌣—⌣—⌣, —⌣⌣—⌣, ⌣⌣⌣⌣—⌣, (2)
⌣——⌣⌣, —⌣—⌣⌣, ⌣⌣⌣—⌣⌣, ——⌣⌣⌣,
⌣⌣—⌣⌣⌣, ⌣—⌣⌣⌣⌣, —⌣⌣⌣⌣⌣, ⌣⌣⌣⌣⌣⌣⌣.

In this way Indian prosodists were led to discover the Fibonacci sequence, as we have observed in Section 1.2.8.

Moreover, the anonymous author of *Prākṛta Paiṅgala* (c. 1320) discovered elegant algorithms for ranking and unranking with respect to m-beat rhythms. To find the kth pattern, one starts by writing down m ⌣s, then expresses the difference $d = F_{m+1} - k$ as a sum of Fibonacci numbers $F_{j_1} + \cdots + F_{j_t}$; here F_{j_1} is the largest Fibonacci number that is $\leq d$ and F_{j_2} is the largest $\leq d - F_{j_1}$, etc., continuing until the remainder is zero. Then beats $j-1$ and j are to be changed from ⌣⌣ to —, for $j = j_1, \ldots, j_t$. For example, to get the 5th element of (2) we compute $21 - 5 = 16 = 13 + 3 = F_7 + F_4$; the answer is ⌣⌣—⌣—.

A few years later, Nārāyaṇa Paṇḍita treated the more general problem of finding all compositions of m whose parts are $\leq q$, where q is *any* given positive integer. As a consequence he discovered the qth-order Fibonacci sequence 5.4.2–(4), which was destined to be used 600 years later in polyphase sorting; he also developed the corresponding ranking and unranking algorithms. [See Parmanand Singh, *Historia Mathematica* **12** (1985), 229–244, and exercise 16.]

Piṅgala gave special code-names to all the three-syllable meters,

——— = म (m), ——⌣ = त (t),
⌣—— = य (y), ⌣—⌣ = ज (j),
—⌣— = र (r), —⌣⌣ = भ (bh), (3)
⌣⌣— = स (s), ⌣⌣⌣ = न (n),

and students of Sanskrit have been expected to memorize them ever since. Somebody long ago devised a clever way to recall these codes, by inventing the nonsense word *yamātārājabhānasalagām* (यमाताराजभानसलगाम्); the point is that the ten syllables of this word can be written

ya mā tā rā ja bhā na sa la gām
⏑ — — — ⏑ — ⏑ ⏑ ⏑ — (4)

and each three-syllable pattern occurs just after its code name. The origin of yamā . . . lagām is obscure, but Subhash Kak [*Indian J. History of Science* **35** (2000), 123–127] has traced it back at least to C. P. Brown's *Sanskrit Prosody* (1869), page 28; thus it qualifies as the earliest known appearance of a "de Bruijn cycle" that encodes binary n-tuples.

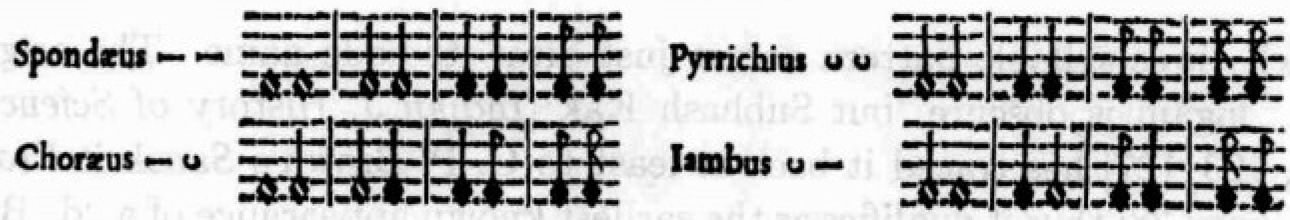
Meanwhile, in Europe. In a similar way, classic Greek poetry was based on groups of short and/or long syllables called "metrical feet," analogous to bars of music. Each basic type of foot acquired a Greek name; for example, two short syllables '⏑⏑' were called a *pyrrhic*, and two long syllables '——' were called a *spondee*, because those rhythms were used respectively in a song of war (πυρρίχη) or a song of peace (σπονδαί). Greek names for metric feet were soon assimilated into Latin and eventually into modern languages, including English:

⏑	arsis	⏑⏑⏑⏑	proceleusmatic	
—	thesis	⏑⏑⏑—	fourth pæon	
		⏑⏑—⏑	third pæon	
⏑⏑	pyrrhic	⏑⏑——	minor ionic	
⏑—	iambus	⏑—⏑⏑	second pæon	
—⏑	trochee	⏑—⏑—	diiambus	
——	spondee	⏑——⏑	antispast	
		⏑———	first epitrite	(5)
⏑⏑⏑	tribrach	—⏑⏑⏑	first pæon	
⏑⏑—	anapest	—⏑⏑—	choriambus	
⏑—⏑	amphibrach	—⏑—⏑	ditrochee	
⏑——	bacchius	—⏑——	second epitrite	
—⏑⏑	dactyl	——⏑⏑	major ionic	
—⏑—	amphimacer	——⏑—	third epitrite	
——⏑	palimbacchius	———⏑	fourth epitrite	
———	molossus	————	dispondee	

Alternative names, like "choree" instead of "trochee," or "cretic" instead of "amphimacer," were also in common use. Moreover, by the time Diomedes wrote his Latin grammar (approximately A.D. 375), each of the 32 *five*-syllable feet had acquired at least one name. Diomedes also pointed out the relation between complementary patterns; he stated for example that tribrach and molossus are "*contrarius*," as are amphibrach and amphimacer. But he also regarded dactyl as the contrary of anapest, and bacchius as the contrary of palimbacchius, although the literal meaning of *palimbacchius* is actually "reverse bacchius." Greek prosodists had no standard order in which to list the individual possibilities, and

the form of the names makes it clear that no connection to a radix-two number system was contemplated. [See H. Keil, *Grammatici Latini* **1** (1857), 474–482; W. von Christ, *Metrik der Griechen und Römer* (1879), 78–79.]

Surviving fragments of a work by Aristoxenus called *Elements of Rhythm* (c. 325 B.C.) show that the same terminology was applied also to music. And indeed, the same traditions lived on after the Renaissance; for example, we find



on page 32 of Athanasius Kircher's *Musurgia Universalis* **2** (Rome: 1650), and Kircher went on to describe all of the three-note and four-note rhythms of (5).

Early lists of permutations. We've traced the history of formulas for counting permutations in Section 5.1.2; but nontrivial *lists* of permutations were not published until hundreds of years after the formula $n!$ was discovered. The first such tabulation currently known was compiled by the Italian physician Shabbetai Donnolo in his commentary on the kabbalistic *Sefer Yetzirah*, written in A.D. 946. Table 1 shows his list for $n = 5$ as it was subsequently printed in Warsaw (1884). (The Hebrew letters in this table are typeset in a rabbinical font traditionally used for commentaries; notice that the letter מ changes its shape to ם when it appears at the left end of a word.) Donnolo went on to list 120 permutations of the six-letter word שבתכמו, all beginning with shin (ש); then he noted that 120 more could be obtained with each of the other five letters in front, making 720 in all. His lists involved groupings of six permutations, but in a haphazard fashion that led him into error (see exercise 4). Although he knew how many permutations there were supposed to be, and how many should start with a given letter, he evidently didn't have an algorithm for generating them.
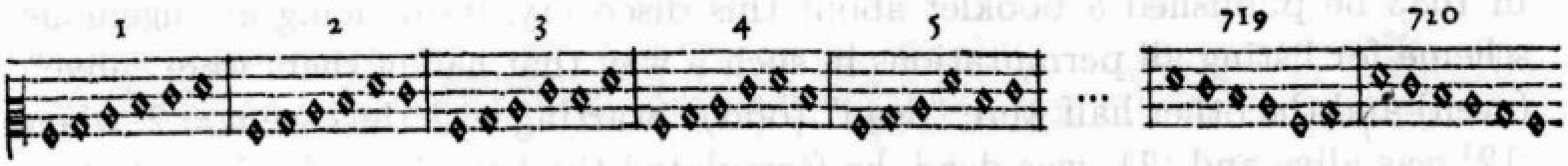
Table 1

A MEDIEVAL LIST OF PERMUTATIONS

דברים , דבירם , דבימר, דבמיר, דבמרי, דברמי, דרבים , דרבמי, דרימב , דריבם, דרמבי, דרמיב,
דיברם, דירבם , דירמב, דיבמר, דימרב, דימבר, דמביר, דמברי, דמרבי, דמריב, דמיבר, דמירב,
בדימר, בדירם, בדרמי, בדרים , בדמרי, בדמיר, בירמד, בירדם, בימדר, בימרד , בידמר,
במדרי , במדיר, במריד , במרדי, במדיר, במירד , בדמיר, בדמרי, בדרמי, בדרים, בדימר , בדירם,
רימדב , רימבד, רידבם, רידמב, ריבדם , ריבמד, רדמבי, רדבמי , רדיבם, רדימב, רדבים , רדמיב,
רבמדי, רבדמי, רבדים, רבידם , רבימד, רבמיד, רמידב , רמיבד, רמדיב, רמביד , רמבדי, רמדבי,
ימדרב, ימרדב, ימבדר, ימברד, ימרבד , ימדבר, ידרבם, ידברם, ידבמר, ידמבר, ידרמב, ידמרב,
יבדרם , יבדמר, יברדם, יברמד, יבמדר, יבמרד, ירדבם , ירדמב , ירמבד, ירבמד, ירבדם , ירדמב,
מדברי, מדרבי, מדביר , מדיבר, מדירב, מדריב, מבדיר , מבידר, מבריד, מבירד, מברדי, מבדרי,
מרדבי, מרבדי, מרדיב, מרביד , מריבד, מרידב, מידבר, מידרב, מיברד, מיבדר, מירדב , מירבד

A complete list of all 720 permutations of $\{a, b, c, d, e, f\}$ appeared on pages 668–671 of Jeremias Drexel's *Orbis Phaëthon* (Munich: 1629; also on pages 526–531 of the Cologne edition in 1631). He offered it as proof that a man with six guests could seat them differently at lunch and dinner every day for a year —

altogether 360 days, because there were five days of fasting during Holy Week. Shortly afterwards, Marin Mersenne exhibited all 720 permutations of the six tones {ut, re, mi, fa, sol, la}, on pages 111–115 of his *Traitez de la Voix et des Chants* (Volume 2 of *Harmonie Universelle*, 1636); then on pages 117–128 he presented the same data in musical notation:



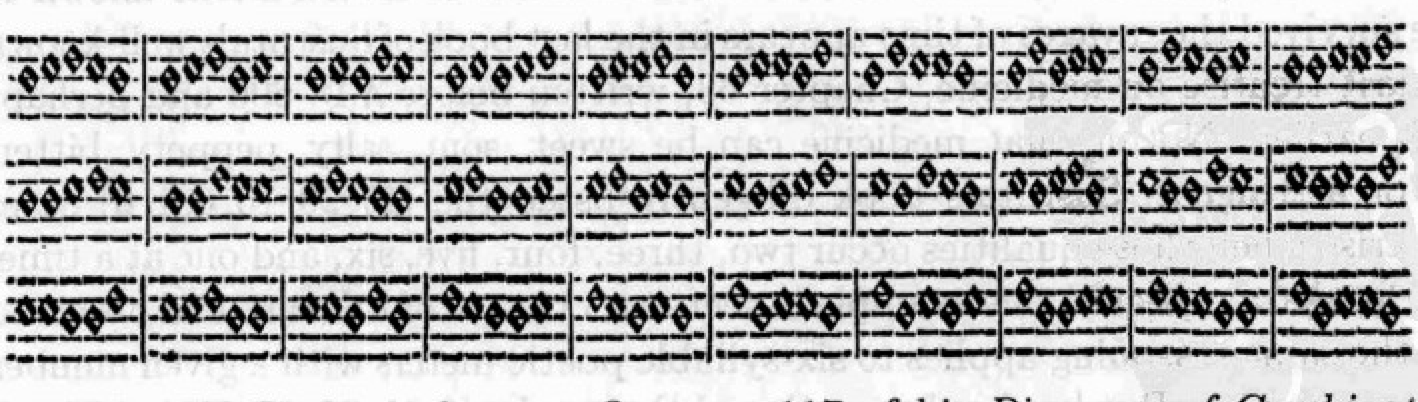
Drexel's table was organized lexicographically by columns; Mersenne's tables were lexicographic with respect to the order ut < re < mi < fa < sol < la, beginning with "ut,re,mi,fa,sol,la" and ending with "la,sol,fa,mi,re,ut." Mersenne also prepared a "grand et immense" manuscript that listed all 40,320 permutations of *eight* notes on 672 folio pages, followed by ranking and unranking algorithms [Bibliothèque nationale de France, Fonds Français, no. 24256].

We saw in Section 7.2.1.2 that the important idea of plain changes, Algorithm 7.2.1.2P, was invented in England a few years later.

Methods for listing all permutations of a multiset with *repeated* elements were often misunderstood by early authors. For example, when Bhāskara exhibited the permutations of {4, 5, 5, 5, 8} in section 271 of his *Līlāvatī* (c. 1150), he gave them in the following order:

४८५५५	८४५५५	५४८५५	५८४५५	५५४८५	
५५८४५	५५५४८	५५५८४	४५८५५	४५५८५	(6)
४५५५८	८५४५५	८५५४५	८५५५४	५४५८५	
५८५४५	५५४५८	५५८५४	५४५५८	५८५५४	

Mersenne used a slightly more sensible but not completely systematic order on page 131 of his book when he listed sixty anagrams of the Latin name IESVS. When Athanasius Kircher wanted to illustrate the 30 permutations of a five-note melody on pages 10 and 11 of *Musurgia Universalis* 2 (1650), this lack of a system got him into trouble (see exercise 5):


(7)

But John Wallis knew better. On page 117 of his *Discourse of Combinations* (1685) he correctly listed the 60 anagrams of "messes" in lexicographic order, if we let m < e < s; and on page 126 he recommended respecting alphabetic order "that we may be the more sure, not to miss any."

We will see later that the Indian pundits Śārṅgadeva and Nārāyaṇa had already developed a theory of permutation generation in the 13th and 14th centuries, although their work was ahead of its time and remained obscure.

Seki's list. Takakazu Seki (1642–1708) was a charismatic teacher and researcher who revolutionized the study of mathematics in 17th-century Japan. While he was studying the elimination of variables from simultaneous homogeneous equations, he was led to expressions such as $a_1b_2 - a_2b_1$ and $a_1b_2c_3 - a_1b_3c_2 + a_2b_3c_1 - a_2b_1c_3 + a_3b_1c_2 - a_3b_2c_1$, which we now recognize as *determinants*. In 1683 he published a booklet about this discovery, introducing an ingenious scheme for listing all permutations in such a way that half of them were "alive" (even) and the other half were "dead" (odd). Starting with the case $n = 2$, when '12' was alive and '21' was dead, he formulated the following rules for $n > 2$:

1) Take every live permutation for $n-1$, increase all its elements by 1, and insert 1 in front. This rule produces $(n-1)!/2$ "basic permutations" of $\{1, \ldots, n\}$.
2) From each basic permutation, form $2n$ others by rotation and reflection:

$$a_1a_2 \ldots a_{n-1}a_n,\ a_2 \ldots a_{n-1}a_na_1,\ \ldots,\ a_na_1a_2 \ldots a_{n-1}; \tag{8}$$
$$a_na_{n-1} \ldots a_2a_1,\ a_1a_na_{n-1} \ldots a_2,\ \ldots,\ a_{n-1} \ldots a_2a_1a_n. \tag{9}$$

If n is odd, those in the first row are alive and those in the second are dead; if n is even, those in each row are alternatively alive, dead, ..., alive, dead.

For example, when $n = 3$ the only basic permutation is 123. Thus 123, 231, 312 are alive while 321, 132, 213 are dead, and we've successfully generated the six terms of a 3×3 determinant. The basic permutations for $n = 4$ are 1234, 1342, 1423; and from, say, 1342 we get a set of eight, namely

$$+1342 - 3421 + 4213 - 2134 + 2431 - 1243 + 3124 - 4321, \tag{10}$$

alternately alive (+) and dead (−). A 4×4 determinant therefore includes the terms $a_1b_3c_4d_2 - a_3b_4c_2d_1 + \cdots - a_4b_3c_2d_1$ and sixteen others.

Seki's rule for permutation generation is quite pretty, but unfortunately it has a serious problem: It doesn't work when $n > 4$. His error seems to have gone unrecognized for hundreds of years. [See Y. Mikami, *The Development of Mathematics in China and Japan* (1913), 191–199; *Takakazu Seki's Collected Works* (Osaka: 1974), 18–20, 八五一 . . . 一四一; and exercises 7–8.]

Lists of combinations. The earliest exhaustive list of *combinations* known to have survived the ravages of time appears in the last book of Suśruta's well-known Sanskrit treatise on medicine, Chapter 63, written before A.D. 600 and perhaps much earlier. Noting that medicine can be sweet, sour, salty, peppery, bitter, and/or astringent, Suśruta's book diligently listed the (15, 20, 15, 6, 1, 6) cases that arise when those qualities occur two, three, four, five, six, and one at a time.

Bhāskara repeated this example in sections 110–114 of *Līlāvatī*, and observed that the same reasoning applies to six-syllable poetic meters with a given number of long syllables. But he simply mentioned the totals, (6, 15, 20, 15, 6, 1), without listing the combinations themselves. In sections 274 and 275, he observed that the numbers $(n)(n-1) \ldots (n-k+1)/(k(k-1) \ldots (1))$ enumerate *compositions* (that is, ordered partitions) as well as combinations; again he gave no list.

To avoid prolixity this is treated in a brief manner;
for the science of calculation is an ocean without bounds.
— *Bhāskara* (c. 1150)

An isolated but interesting list of combinations appeared in the remarkable algebra text *Al-Bāhir fi'l-ḥisāb* (*The Shining Book of Calculation*), written by al-Samaw'al of Baghdad when he was only 19 years old (1144). In the closing part of that work he presented a list of $\binom{10}{6} = 210$ simultaneous linear equations in 10 unknowns:

Al-Samaw'al's Arabic original			Equivalent modern notation
٦٥	٦٥٤٣٢١	ا	(1) $x_1 + x_2 + x_3 + x_4 + x_5 + x_6 = 65$
٧٠	٧٥٤٣٢١	ب	(2) $x_1 + x_2 + x_3 + x_4 + x_5 + x_7 = 70$
٧٥	٨٥٤٣٢١	جـ	(3) $x_1 + x_2 + x_3 + x_4 + x_5 + x_8 = 75$
⋮	⋮		⋮
٩١	١٠٩٨٧٦٤	زط	(209) $x_4 + x_6 + x_7 + x_8 + x_9 + x_{10} = 91$
١٠٠	١٠٩٨٧٦٥	زي	(210) $x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} = 100$

(11)

Each combination of ten things taken six at a time yielded one of his equations. His purpose was evidently to demonstrate that over-determined equations can still have a unique solution — which in this case was $(x_1, x_2, \ldots, x_{10}) = (1, 4, 9, 16, 25, 10, 15, 20, 25, 5)$. [Salah Ahmad and Roshdi Rashed, *Al-Bāhir en Algèbre d'As-Samaw'al* (Damascus: 1972), 77–82, ٢٤٨–٢٣١.]

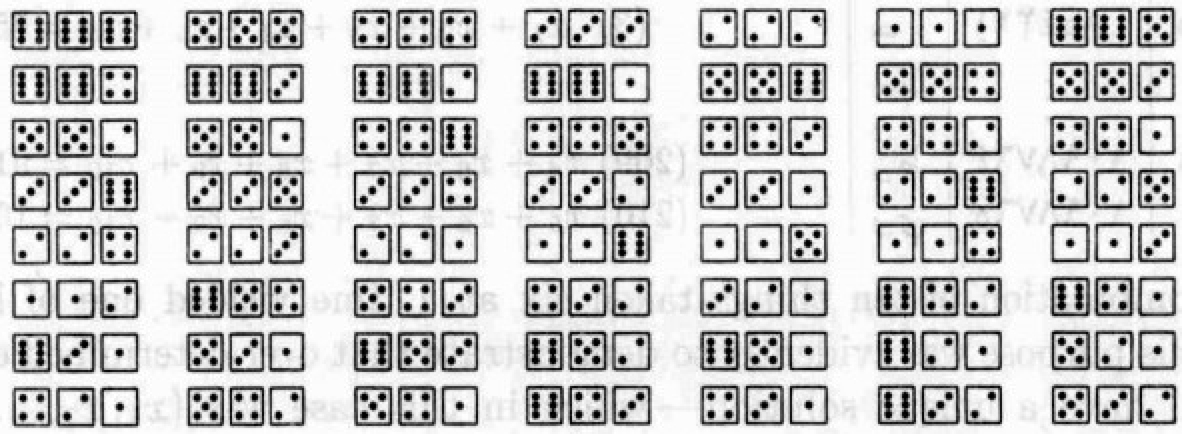
Rolling dice. Some glimmerings of elementary combinatorics arose also in medieval Europe, especially in connection with the question of listing all possible outcomes when three dice are thrown. There are, of course, $\binom{8}{3} = 56$ ways to choose 3 things from 6 when repetitions are allowed. Gambling was officially prohibited; yet these 56 ways became rather well known. In about A.D. 965, Bishop Wibold of Cambrai in northern France devised a game called Ludus Clericalis, so that members of the clergy could enjoy rolling dice while remaining pious. His idea was to associate each possible roll with one of 56 virtues, according to the following table:

⚀⚀⚀	love	⚀⚂⚅	perseverance	⚁⚂⚄	hospitality	⚂⚃⚅	mortification
⚀⚀⚁	faith	⚀⚃⚃	kindness	⚁⚂⚅	economy	⚂⚄⚄	innocence
⚀⚀⚂	hope	⚀⚃⚄	modesty	⚁⚃⚃	patience	⚂⚄⚅	contrition
⚀⚀⚃	justice	⚀⚃⚅	resignation	⚁⚃⚄	zeal	⚂⚅⚅	confession
⚀⚀⚄	prudence	⚀⚄⚄	gentleness	⚁⚃⚅	poverty	⚃⚃⚃	maturity
⚀⚀⚅	temperance	⚀⚄⚅	generosity	⚁⚄⚄	softness	⚃⚃⚄	solicitude
⚀⚁⚁	courage	⚀⚅⚅	wisdom	⚁⚄⚅	virginity	⚃⚃⚅	constancy
⚀⚁⚂	peace	⚁⚁⚁	remorse	⚁⚅⚅	respect	⚃⚄⚄	intelligence
⚀⚁⚃	chastity	⚁⚁⚂	joy	⚂⚂⚂	piety	⚃⚄⚅	sighing
⚀⚁⚄	mercy	⚁⚁⚃	sobriety	⚂⚂⚃	indulgence	⚃⚅⚅	weeping
⚀⚁⚅	obedience	⚁⚁⚄	satisfaction	⚂⚂⚄	prayer	⚄⚄⚄	cheerfulness
⚀⚂⚂	fear	⚁⚁⚅	sweetness	⚂⚂⚅	affection	⚄⚄⚅	compassion
⚀⚂⚃	foresight	⚁⚂⚂	cleverness	⚂⚃⚃	judgment	⚄⚅⚅	self-control
⚀⚂⚄	discretion	⚁⚂⚃	simplicity	⚂⚃⚄	vigilance	⚅⚅⚅	humility

Players took turns, and the first to roll each virtue acquired it. After all possibilities had arisen, the most virtuous player won. Wibold noted that love (*caritas*) is the best virtue of all. He gave a complicated scoring system by which two virtues could be combined if the sum of pips on all six of their dice was 21; for

example, love + humility or chastity + intelligence could be paired in this way, and such combinations ranked above any individual virtue. He also considered more complex variants of the game in which vowels appeared on the dice instead of spots, so that virtues could be claimed if their vowels were thrown.

Wibold's table of virtues was presented in lexicographic order, as above, when it was first described by Baldéric in his *Chronicon Cameracense*, about 150 years later. [*Patrologia Latina* **134** (Paris: 1884), 1007–1016.] But another medieval manuscript presented the possible dice rolls in quite a different order:



(12)

In this case the author knew how to deal with repeated values, but had a very complicated, ad hoc way to handle the cases in which all dice were different. [See D. R. Bellhouse, *International Statistical Review* **68** (2000), 123–136.]

An amusing poem entitled "Chaunce of the Dyse," attributed to John Lydgate, was written in the early 1400s for use at parties. Its opening verses invite each person to throw three dice; then the remaining verses, which are indexed in decreasing lexicographic order from ⚅⚅⚅ to ⚅⚅⚄ to ··· to ⚀⚀⚀, give 56 character sketches that light-heartedly describe the thrower. [The full text was published by E. P. Hammond in *Englische Studien* **59** (1925), 1–16; a translation into modern English would be desirable.]

I pray to god that euery wight may caste
Vpon three dyse ryght as is in hys herte
Whether he be rechelesse or stedfaste
So moote he lawghen outher elles smerte
He that is gilty his lyfe to converte
They that in trouthe haue suffred many a throwe
Moote ther chaunce fal as they moote be knowe.

— *The Chaunce of the Dyse* (c. 1410)

Ramon Llull. Significant ripples of combinatorial concepts also emanated from an energetic and quixotic Catalan poet, novelist, encyclopedist, educator, mystic, and missionary named Ramon Llull (c. 1232–1316). Llull's approach to knowledge was essentially to identify basic principles and then to contemplate combining them in all possible ways.

For example, one chapter in his *Ars Compendiosa Inveniendi Veritatem* (c. 1274) began by enumerating sixteen attributes of God: Goodness, greatness, eternity, power, wisdom, love, virtue, truth, glory, perfection, justice, generosi y, mercy, humility, sovereignty, and patience. Then Llull wrote $\binom{16}{2} = 120$ short essays of about 80 words each, considering God's goodness as related to greatness,

God's goodness as related to eternity, and so on, ending with God's sovereignty as related to patience. In another chapter he considered seven virtues (faith, hope, charity, justice, prudence, fortitude, temperance) and seven vices (gluttony, lust, greed, sloth, pride, envy, anger), with $\binom{14}{2} = 91$ subchapters to deal with each pair in turn. Other chapters were systematically divided in a similar way, into $\binom{8}{2} = 28$, $\binom{15}{2} = 105$, $\binom{4}{2} = 6$, and $\binom{16}{2} = 120$ subsections. (One wonders what might have happened if he had been familiar with Wibold's list of 56 virtues: would he have produced commentaries on all $\binom{56}{2} = 1540$ of their pairs?)

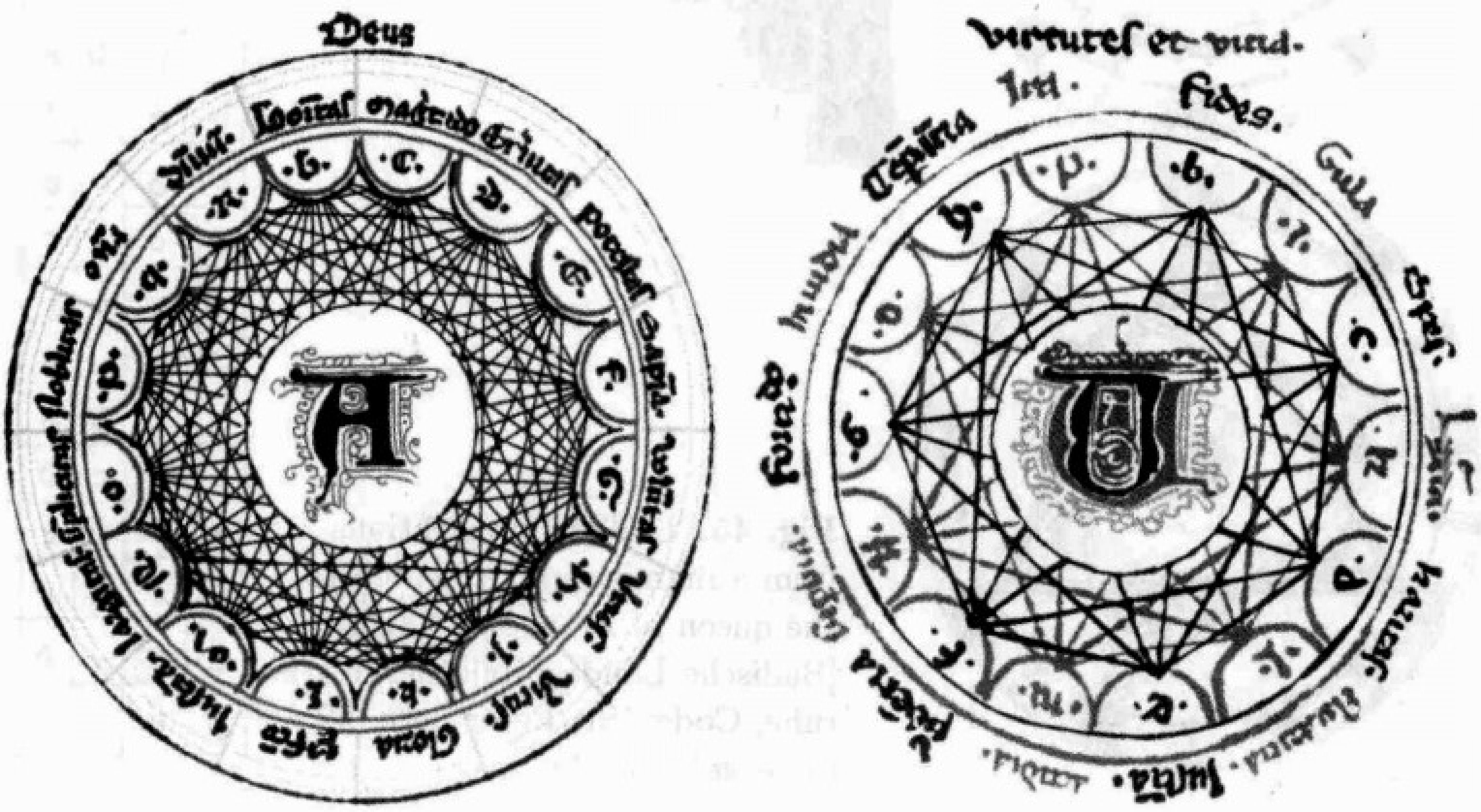


Fig. 44. Illustrations in a manuscript presented by Ramon Llull to the doge of Venice in 1280. [From his *Ars Demonstrativa*, Biblioteca Marciana, VI 200, folio 3ᵛ.]

Llull illustrated his methodology by drawing circular diagrams like those in Figure 44. The left-hand circle in this illustration, *Deus*, names sixteen divine attributes — essentially the same sixteen listed earlier, except that love (*amor*) was now called will (*voluntas*), and the final four were now respectively simplicity, rank, mercy, and sovereignty. Each attribute is assigned a code letter, and the illustration depicts their interrelations as the complete graph K_{16} on vertices (B, C, D, E, F, G, H, I, K, L, M, N, O, P, Q, R). The right-hand figure, *virtutes et vitia,* shows the seven virtues (b, c, d, e, f, g, h) interleaved with the seven vices (i, k, l, m, n, o, p); in the original manuscript virtues appeared in blue ink while vices appeared in red. Notice that in this case his illustration depicted two independent complete graphs K_7, one of each color. (He no longer bothered to compare each individual virtue with each individual vice, since every virtue was clearly better than every vice.)

Llull used the same approach to write about medicine: Instead of juxtaposing theological concepts, his *Liber Principiorum Medicinæ* (c. 1275) considered combinations of symptoms and treatments. And he also wrote books

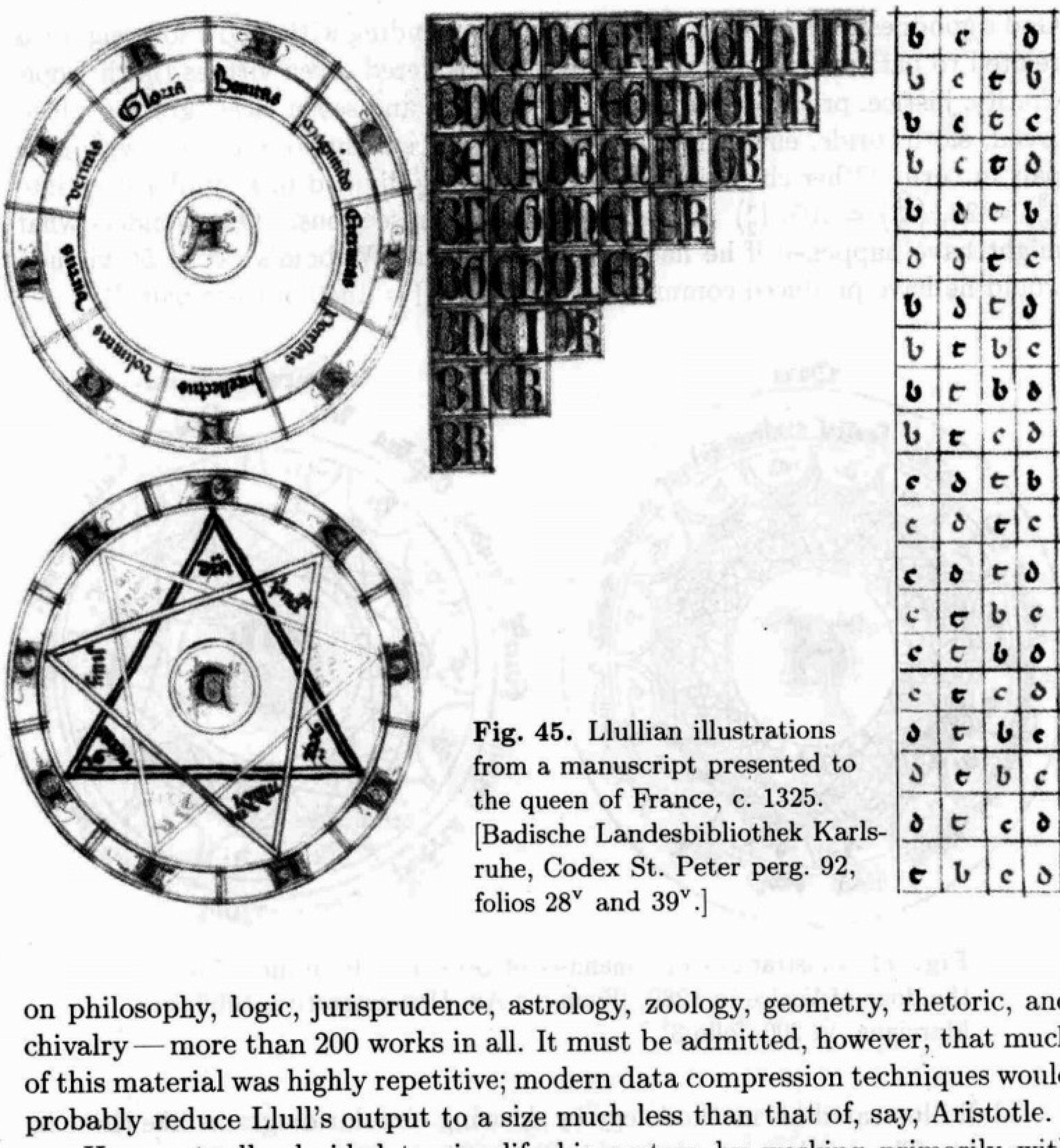


Fig. 45. Llullian illustrations from a manuscript presented to the queen of France, c. 1325. [Badische Landesbibliothek Karlsruhe, Codex St. Peter perg. 92, folios 28$^{\text{v}}$ and 39$^{\text{v}}$.]

on philosophy, logic, jurisprudence, astrology, zoology, geometry, rhetoric, and chivalry — more than 200 works in all. It must be admitted, however, that much of this material was highly repetitive; modern data compression techniques would probably reduce Llull's output to a size much less than that of, say, Aristotle.

He eventually decided to simplify his system by working primarily with groups of nine things. See, for example, Fig. 45, where circle A now lists only the first nine of God's attributes (B, C, D, E, F, G, H, I, K). The $\binom{9}{2} = 36$ associated pairs (BC, BD, ..., IK) appear in the stairstep chart at the right of that circle. By adding two more virtues, namely patience and compassion — as well as two more vices, namely lying and inconsistency — he could treat virtues vis-à-vis virtues and vices vis-à-vis vices with the same chart. He also proposed using the same chart to carry out an interesting scheme for voting, in an election with nine candidates [see I. McLean and J. London, *Studia Lulliana* **32** (1992), 21–37].

The encircled triangles at the lower left of Fig. 45 illustrate another key aspect of Llull's approach. Triangle (B, C, D) stands for (difference, concordance, contrariness); triangle (E, F, G) stands for (beginning, middle, ending); and triangle (H, I, K) stands for (greater, equal, less). These three interleaved appearances of K_3 represent three kinds of three-valued logic. Llull had experimented earlier with other such triplets, notably '(true, unknown, false)'. We can get an idea

of how he used the triangles by considering how he dealt with combinations of the four basic elements (earth, air, fire, water): All four elements are different; earth is concordant with fire, which concords with air, which concords with water, which concords with earth; earth is contrary to air, and fire is contrary to water; these considerations complete an analysis with respect to triangle (B, C, D). Turning to triangle (E, F, G), he noted that various processes in nature begin with one element dominating another; then a transition or middle state occurs, until a goal is reached, like air becoming warm. For triangle (H, I, K) he said that in general we have fire > air > water > earth with respect to their "spheres," their "velocities," and their "nobilities"; nevertheless we also have, for example, air > fire with respect to supporting life, while air and fire have equal value when they are working together.

Llull provided the vertical table at the right of Fig. 45 as a further aid. (See exercise 11 below.) He also introduced movable concentric wheels, labeled with the letters (B, C, D, E, F, G, H, I, K) and with other names, so that many things could be contemplated simultaneously. In this way a faithful practitioner of the Llullian art could be sure to have all the bases covered. [Llull may have seen similar wheels that were used in nearby Jewish communities; see M. Idel, *J. Warburg and Courtauld Institutes* **51** (1988), 170–174 and plates 16–17.]

Several centuries later, Athanasius Kircher published an extension of Llull's system as part of a large tome entitled *Ars Magna Sciendi sive Combinatoria* (Amsterdam: 1669), with five movable wheels accompanying page 173 of that book. Kircher also extended Llull's repertoire of complete graphs K_n by providing illustrations of complete *bipartite* graphs $K_{m,n}$; for example, Fig. 46 is taken from page 171 of Kircher's book, and his page 170 contains a glorious picture of $K_{18,18}$.

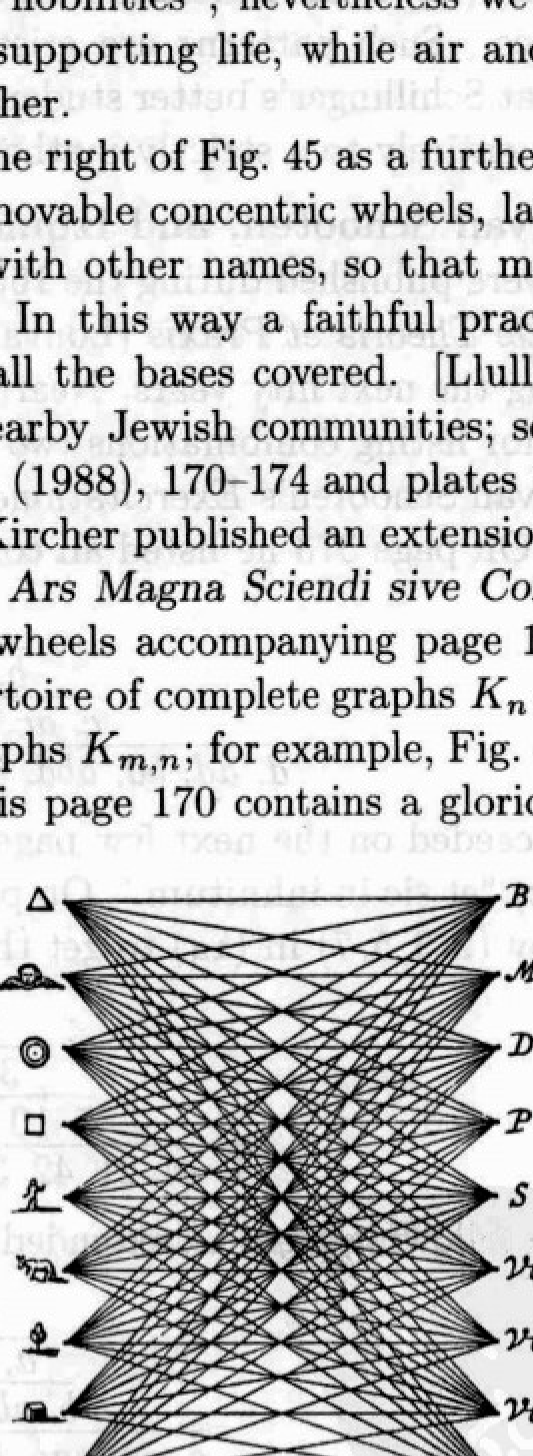


Fig. 46. $K_{9,9}$ as presented by Athanasius Kircher in 1669.

It is an investigative and inventive art.
When ideas are combined in all possible ways,
the new combinations start the mind thinking along novel channels
and one is led to discover fresh truths and arguments.
— MARTIN GARDNER, *Logic Machines and Diagrams* (1958)

The most extensive modern development of Llull-like methods is perhaps *The Schillinger System of Musical Composition* by Joseph Schillinger (New York:

Carl Fischer, 1946), a remarkable two-volume work that presents theories of rhythm, melody, harmony, counterpoint, composition, orchestration, etc., from a combinatorial perspective. On page 56, for example, Schillinger lists the 24 permutations of $\{a, b, c, d\}$ in the Gray-code order of plain changes (Algorithm 7.2.1.2P); then on page 57 he applies them not to pitches but rather to rhythms, to the durations of notes. On page 364 he exhibits the symmetrical cycle

$$(2, 0, 3, 4, 2, 5, 6, 4, 0, 1, 6, 2, 3, 1, 4, 5, 3, 6, 0, 5, 1), \qquad (13)$$

a universal cycle of 2-combinations for the seven objects $\{0, 1, 2, 3, 4, 5, 6\}$; in other words, (13) is an Eulerian trail in K_7: All $\binom{7}{2} = 21$ pairs of digits occur exactly once. Such patterns are grist for a composer's mill. But we can be grateful that Schillinger's better students (like George Gershwin) did not commit themselves entirely to a strictly mathematical sense of aesthetics.

Tacquet, van Schooten, and Izquierdo. Three additional books related to our story were published during the 1650s. André Tacquet wrote a popular text, *Arithmeticæ Theoria et Praxis* (Louvain: 1656), that was reprinted and revised often during the next fifty years. Near the end, on pages 376 and 377, he gave a procedure for listing combinations two at a time, then three at a time, etc.

Frans van Schooten's *Exercitationes Mathematicæ* (Leiden: 1657) was more advanced. On page 373 he listed all combinations in an appealing layout

$$\begin{array}{c} a \\ \hline b.\ ab \\ \hline c.\ ac.\ bc.\ abc \\ \hline d.\ ad.\ bd.\ abd.\ cd.\ acd.\ bcd.\ abcd \end{array} \qquad (14)$$

and he proceeded on the next few pages to extend this pattern to the letters e, f, g, h, i, k, "et sic in infinitum." On page 376 he observed that one can replace (a, b, c, d) by $(2, 3, 5, 7)$ in (14) to get the divisors of 210 that exceed unity:

$$\begin{array}{c} 2 \\ \hline 3\ 6 \\ \hline 5\ 10\ 15\ 30 \\ \hline 7\ 14\ 21\ 42\ 35\ 70\ 105\ 210 \end{array} \qquad (15)$$

And on the following page he extended the idea to

$$\begin{array}{c} a \\ \hline a.\ aa \\ \hline b.\ ab.\ aab \\ \hline c.\ ac.\ aac.\ bc.\ abc.\ aabc \end{array} \qquad (16)$$

thereby allowing two a's. He didn't really understand this extension, though; his next example

$$\begin{array}{c} a \\ \hline a.\ aa \\ \hline a.\ aaa \\ \hline b.\ ab.\ aab.\ aaab \\ \hline b.\ bb.\ abb.\ aabb.\ aaabb \end{array} \qquad (17)$$

was botched, indicating the limits of his knowledge at the time. (See exercise 13.)

On page 411 van Schooten observed that the weights $(a, b, c, d) = (1, 2, 4, 8)$ could be assigned in (14), leading to

$$\begin{array}{c} \underline{1} \\ \underline{2\ \ 3} \\ \underline{4\ \ 5\ \ 6\ \ 7} \\ 8\ \ 9\ \ 10\ \ 11\ \ 12\ \ 13\ \ 14\ \ 15 \end{array} \tag{18}$$

after addition. But he didn't see the connection with radix-2 arithmetic.

Sebastián Izquierdo's two-volume work *Pharus Scientiarum* (Lyon: 1659), "The Lighthouse of Science," included a nicely organized discussion of combinatorics entitled Disputatio 29, *De Combinatione*. He gave a detailed discussion of four key parts of Stanley's Twelvefold Way, namely the n-tuples, n-variations, n-multicombinations, and n-combinations of m objects that appear in the first two rows and the first two columns of Table 7.2.1.4–1.

In Sections 81–84 of *De Combinatione* he listed all combinations of m letters taken n at a time, for $2 \le n \le 5$ and $n \le m \le 9$, always in lexicographic order; he also tabulated them for $m = 10$ and 20 in the cases $n = 2$ and 3. But when he listed the $m^{\underline{n}}$ *variations* of m things taken n at a time, he chose a more complicated ordering (see exercise 14).

Izquierdo was first to discover the formula $\binom{m+n-1}{n}$ for combinations of m things taken n at a time with unlimited repetition; this rule appeared in §48–§51 of his work. But in §105, when he attempted to list all such combinations in the case $n = 3$, he didn't know that there was a simple way to do it. In fact, his listing of the 56 cases for $m = 6$ was rather like the old, awkward ordering of (12).

Combinations with repetition were not well understood until James Bernoulli's *Ars Conjectandi*, "The Art of Guessing," came out in 1713. In Part 2, Chapter 5, Bernoulli simply listed the possibilities in lexicographic order, and showed that the formula $\binom{m+n-1}{n}$ follows by induction as an easy consequence. [Niccolò Tartaglia had, incidentally, come close to discovering this formula in his *General trattato di numeri, et misure* **2** (Venice: 1556), 17r and 69v; so had the Maghrebi mathematician Ibn Mun'im in his 13th-century *Fiqh al-Ḥisāb*.]

The null case. Before we conclude our discussion of early work on combinations, we should not forget a small yet noble step taken by John Wallis on page 110 of his *Discourse of Combinations* (1685), where he specifically considered the combination of m things taken 0 at a time: "It is manifest, That, if we would *take None*, that is, if we would *leave All*; there can be but one case thereof, what ever be the Number of things exposed." Furthermore, on page 113, he knew that $\binom{0}{0} = 1$: "(for, here, to take all, or to leave all, is but one and the same case.)"

However, when he gave a table of $n!$ for $n \le 24$, he did not go so far as to point out that $0! = 1$, or that there is exactly one permutation of the empty set.

The work of Nārāyaṇa. A remarkable monograph entitled *Gaṇita Kaumudī* ("Lotus Delight of Calculation"), written by Nārāyaṇa Paṇḍita in 1356, has recently become known in detail to scholars outside of India for the first time, thanks to an English translation by Parmanand Singh [*Gaṇita Bhāratī* **20** (1998), 25–82; **21** (1999), 10–73; **22** (2000), 19–85; **23** (2001), 18–82; **24** (2002), 35–98];

see also the Ph.D. thesis of Takanori Kusuba, Brown University (1993). Chapter 13 of Nārāyaṇa's work, subtitled *Aṅka Pāśa* ("Concatenation of Numbers"), was devoted to combinatorial generation. Indeed, although the 97 "sutras" of this chapter were rather cryptic, they presented a comprehensive theory of the subject that anticipated developments in the rest of the world by several hundred years.

For example, Nārāyaṇa dealt with permutation generation in sutras 49–55a, where he gave algorithms to list all permutations of a set in decreasing colex order, together with algorithms to rank a given permutation and to unrank a given serial number. These algorithms had appeared more than a century earlier in the well-known work *Saṅgītaratnākara* ("Jewel-Mine of Music") by Śārṅgadeva, §1.4.60–71, who thereby had essentially discovered the factorial representation of positive integers. Nārāyaṇa went on in sutras 57–60 to extend Śārṅgadeva's algorithms so that general multisets could readily be permuted; for example, he listed the permutations of $\{1,1,2,4\}$ as

$$1124, 1214, 2114, 1142, 1412, 4112, 1241, 2141, 1421, 4121, 2411, 4211,$$

again in decreasing colex order.

Nārāyaṇa's sutras 88–92 dealt with systematic generation of combinations. Besides illustrating the combinations of $\{1,\ldots,8\}$ taken 3 at a time, namely

$$(678, 578, 478, \ldots, 134, 124, 123),$$

he also considered a bit-string representation of these combinations in the reverse order (*increasing* colex), extending a 10th-century method of Bhaṭṭotpala:

$$(11100000, 11010000, 10110000, \ldots, 00010011, 00001011, 00000111).$$

He almost, but not quite, discovered Theorem 7.2.1.3L.

Permutable poetry. Let's turn now to a curious question that attracted the attention of several prominent mathematicians in the seventeenth century, because it sheds considerable light on the state of combinatorial knowledge in Europe at that time. A Jesuit priest named Bernard Bauhuis had composed a famous one-line tribute to the Virgin Mary, in Latin hexameter:

Tot tibi sunt dotes, Virgo, quot sidera cælo. (19)

["Thou hast as many virtues, O Virgin, as there are stars in heaven"; see his *Epigrammatum Libri V* (Cologne: 1615), 49.] His verse inspired Erycius Puteanus, a professor at the University of Louvain, to write a book entitled *Pietatis Thaumata* (Antwerp: 1617), presenting 1022 permutations of Bauhuis's words. For example, Puteanus wrote

107 Tot dotes tibi, quot cælo sunt sidera, Virgo.
270 Dotes tot, cælo sunt sidera quot, tibi Virgo.
329 Dotes, cælo sunt quot sidera, Virgo tibi tot.
384 Sidera quot cælo, tot sunt Virgo tibi dotes. (20)
725 Quot cælo sunt sidera, tot Virgo tibi dotes.
949 Sunt dotes Virgo, quot sidera, tot tibi cælo.
1022 Sunt cælo tot Virgo tibi, quot sidera, dotes.

He stopped at 1022, because 1022 was the number of visible stars in Ptolemy's well-known catalog of the heavens.

The idea of permuting words in this way was well known at the time; such wordplay was what Julius Scaliger had called "Proteus verses" in his *Poetices Libri Septem* (Lyon: 1561), Book 2, Chapter 30. The Latin language lends itself to permutations like (20), because Latin word endings tend to define the function of each noun, making the relative word order much less important to the meaning of a sentence than it is in English. Puteanus did state, however, that he had specifically avoided unsuitable permutations such as

$$\text{Sidera tot cælo, Virgo, quot sunt tibi dotes,} \qquad (21)$$

because they would place an *upper* bound on the Virgin's virtues rather than a lower bound. [See pages 12 and 103 of his book.]

Of course there are $8! = 40{,}320$ ways to permute the words of (19). But that wasn't the point; most of those ways don't "scan." Each of Puteanus's 1022 verses obeyed the strict rules of classical *hexameter*, the rules that had been followed by Greek and Latin poets since the days of Homer and Vergil, namely:

i) Each word consists of syllables that are either long ($-$) or short ($\smile$);
ii) The syllables of each line belong to one of 32 patterns,

$$\left\{\begin{matrix}-\smile\smile\\--\end{matrix}\right\}\left\{\begin{matrix}-\smile\smile\\--\end{matrix}\right\}\left\{\begin{matrix}-\smile\smile\\--\end{matrix}\right\}\left\{\begin{matrix}-\smile\smile\\--\end{matrix}\right\}-\smile\smile\left\{\begin{matrix}-\smile\\--\end{matrix}\right\}. \qquad (22)$$

In other words there are six metrical feet, where each of the first four is either a dactyl or a spondee in the terminology of (5); the fifth foot should be a dactyl, and the last is either trochee or spondee.

The rules for long versus short syllables in Latin poetry are somewhat tricky in general, but the eight words of Bauhuis's verse can be characterized by the following patterns:

$$\text{tot} = -,\ \text{tibi} = \left\{\begin{matrix}\smile\smile\\\smile-\end{matrix}\right\},\ \text{sunt} = -,\ \text{dotes} = --,$$
$$\text{Virgo} = \left\{\begin{matrix}-\smile\\--\end{matrix}\right\},\ \text{quot} = -,\ \text{sidera} = -\smile\smile,\ \text{cælo} = --. \qquad (23)$$

Notice that poets had two choices when they used the words 'tibi' or 'Virgo'. Thus, for example, (19) fits the hexameter pattern

$$\overset{-}{\text{Tot}}\ \overset{\smile}{\text{ti-}}\overset{\smile}{\text{bi}}\quad \overset{-}{\text{sunt}}\ \overset{-}{\text{do-}}\quad \overset{-}{\text{tes,}}\ \overset{-}{\text{Vir-}}\quad \overset{-}{\text{go,}}\ \overset{-}{\text{quot}}\quad \overset{-}{\text{si-}}\overset{\smile}{\text{de-}}\overset{\smile}{\text{ra}}\quad \overset{-}{\text{cæ-}}\overset{-}{\text{lo.}} \qquad (24)$$

(Dactyl, spondee, spondee, spondee, dactyl, spondee; "dum-diddy dum-dum dum-dum dum-dum dum-diddy dum-dum." The commas represent slight pauses, called "cæsuras," when the words are read; they don't concern us here, although Puteanus inserted them carefully into each of his 1022 permutations.)

A natural question now arises: If we permute Bauhuis's words at random, what are the odds that they scan? In other words, how many of the permutations obey rules (i) and (ii), given the syllable patterns in (23)? G. W. Leibniz raised

this question, among others, in his *Dissertatio de Arte Combinatoria* (1666), a work published when he was applying for a position at the University of Leipzig. At this time Leibniz was just 19 years old, largely self-taught, and his understanding of combinatorics was quite limited; for example, he believed that there are 600 permutations of $\{\mathrm{ut}, \mathrm{ut}, \mathrm{re}, \mathrm{mi}, \mathrm{fa}, \mathrm{sol}\}$ and 480 of $\{\mathrm{ut}, \mathrm{ut}, \mathrm{re}, \mathrm{re}, \mathrm{mi}, \mathrm{fa}\}$, and he even stated that (22) represents 76 possibilities instead of 32. [See §5 and §8 in his Problem 6.]

But Leibniz did realize that it would be worthwhile to develop general methods for counting all permutations that are "useful," in situations when many permutations are "useless." He considered several examples of Proteus verses, enumerating some of the simpler ones correctly but making many errors when the words were complicated. Although he mentioned Puteanus's work, he didn't attempt to enumerate the scannable permutations of (19).

A much more successful approach was introduced a few years later by Jean Prestet in his *Élémens des Mathématiques* (Paris: 1675), 342–438. Prestet gave a clear exposition leading to the conclusion that exactly 2196 permutations of Bauhuis's verse would yield a proper hexameter. However, he soon realized that he had forgotten to count quite a few cases — including those numbered 270, 384, and 725 in (20). So he completely rewrote this material when he published *Nouveaux Élémens des Mathématiques* in 1689. Pages 127–133 of Prestet's new book were devoted to showing that the true number of scannable permutations was 3276, almost 50% larger than his previous total.

Meanwhile John Wallis had treated the problem in his *Discourse of Combinations* (London: 1685), 118–119, published as a supplement to his *Treatise of Algebra*. After explaining why he believed the correct number to be 3096, Wallis admitted that he may have overlooked some possibilities and/or counted some cases more than once; "but I do not, at present, discern either the one and other."

An anonymous reviewer of Wallis's work remarked that the true number of metrically correct permutations was actually 2580 — but he gave no proof [*Acta Eruditorum* **5** (1686), 289]. The reviewer was almost certainly G. W. Leibniz himself, although no clue to the reasoning behind the number 2580 has been found among Leibniz's voluminous unpublished notes.

Finally James Bernoulli entered the picture. In his inaugural lecture as Dean of Philosophy at the University of Basel, 1692, he mentioned the tot-tibi enumeration problem and stated that a careful analysis is necessary to obtain the correct answer — which, he said, was 3312(!). His proof appeared posthumously in the first edition of his *Ars Conjectandi* (1713), 79–81. [Those pages were, incidentally, omitted from later editions of that famous book, and from his collected works, because he didn't actually intend them for publication; a proofreader had inserted them by mistake. See *Die Werke von Jakob Bernoulli* **3** (Basel: Birkhäuser, 1975), 78, 98–106, 108, 154–155.]

So who was right? Are there 2196 scannable permutations, or 3276, or 3096, or 2580, or 3312? W. A. Whitworth and W. E. Hartley considered the question anew in *The Mathematical Gazette* **2** (1902), 227–228, where they each presented elegant arguments and concluded that the true total was in fact none of the

above. Their joint answer, 2880, represented the first time that any two mathematicians had independently come to the same conclusion about this problem.

But exercises 21 and 22, below, reveal the truth: Bernoulli is vindicated, and everybody else was wrong. Moreover, a study of Bernoulli's systematic and carefully indented 3-page derivation indicates that he was successful chiefly because he adhered faithfully to a discipline that we now call the *backtrack method.* We shall study the backtrack method thoroughly in Section 7.2.2, where we will also see that the tot-tibi question is readily solved as a special case of the *exact cover problem.*

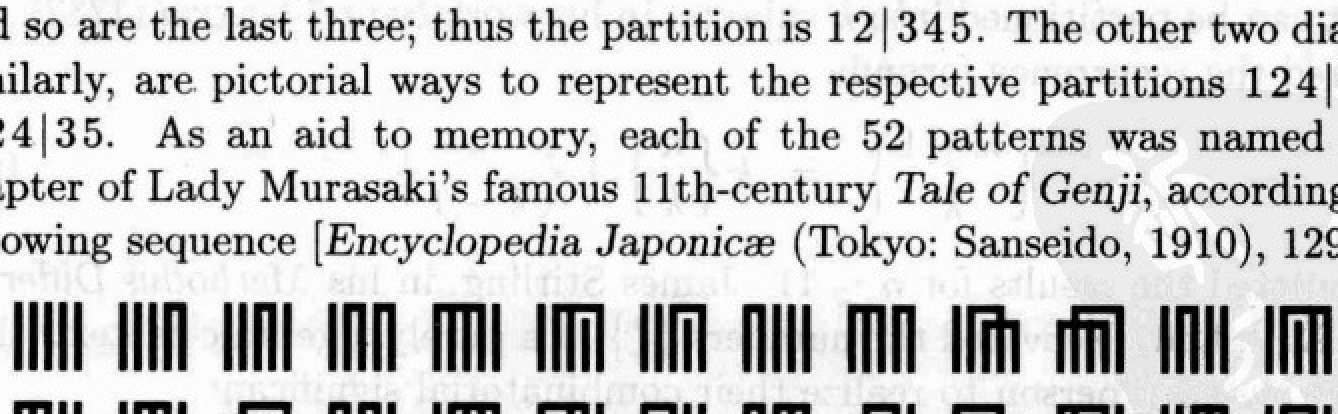
Even the wisest and most prudent people often suffer from what Logicians call insufficient enumeration of cases.
— JAMES BERNOULLI (1692)

Set partitions. The partitions of a set seem to have been studied first in Japan, where a parlor game called *genji-ko* ("Genji incense") became popular among upperclass people about A.D. 1500. The host of a gathering would secretly select five packets of incense, some of which might be identical, and he would burn them one at a time. The guests would try to discern which of the scents were the same and which were different; in other words, they would try to guess which of the $\varpi_5 = 52$ partitions of $\{1, 2, 3, 4, 5\}$ had been chosen by their host.



Fig. 47. Diagrams used to represent set partitions in 16th century Japan. [From a copy in the collection of Tamaki Yano at Saitama University.]

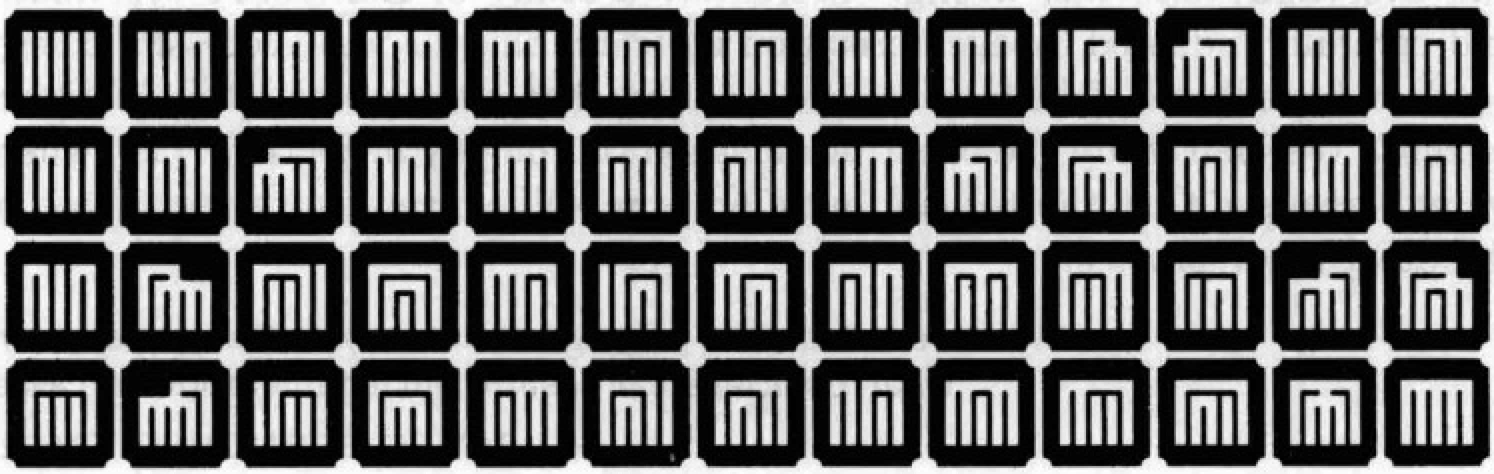
Soon it became customary to represent the 52 possible outcomes by diagrams like those in Fig. 47. For example, the uppermost diagram of that illustration, when read from right to left, would indicate that the first two scents are identical and so are the last three; thus the partition is 12|345. The other two diagrams, similarly, are pictorial ways to represent the respective partitions 124|35 and 1|24|35. As an aid to memory, each of the 52 patterns was named after a chapter of Lady Murasaki's famous 11th-century *Tale of Genji*, according to the following sequence [*Encyclopedia Japonicæ* (Tokyo: Sanseido, 1910), 1299]:



(25)

(Once again, as we've seen in many other examples, the possibilities were not arranged in any particularly logical order.)

The appealing nature of these genji-ko patterns led many families to adopt them as heraldic crests. For example, the following stylized variants of (25) were found in standard catalogs of kimono patterns early in the 20th century:



[See Fumie Adachi, *Japanese Design Motifs* (New York: Dover, 1972), 150–153.]

Early in the 1700s, Takakazu Seki and his students began to investigate the number of set partitions ϖ_n for arbitrary n, inspired by the known result that $\varpi_5 = 52$. Yoshisuke Matsunaga found formulas for the number of set partitions when there are k_j subsets of size n_j for $1 \le j \le t$, with $k_1n_1 + \cdots + k_tn_t = n$ (see the answer to exercise 1.2.5–21). He also discovered the basic recurrence relation 7.2.1.5–(14), namely

$$\varpi_{n+1} \;=\; \binom{n}{0}\varpi_n + \binom{n}{1}\varpi_{n-1} + \binom{n}{2}\varpi_{n-2} + \cdots + \binom{n}{n}\varpi_0, \qquad (26)$$

by which the values of ϖ_n can readily be computed.

Matsunaga's discoveries remained unpublished until Yoriyuki Arima's book *Shūki Sanpō* came out in 1769. Problem 56 of that book asked the reader to solve the equation "$\varpi_n = 678570$" for n; and Arima's answer, worked out in detail (with credit duly given to Matsunaga), was $n = 11$.

Shortly afterwards, Masanobu Saka studied the number $\left\{{n \atop k}\right\}$ of ways that an n-set can be partitioned into k subsets, in his work *Sanpō-Gakkai* (1782). He discovered the recurrence formula

$$\left\{{n+1 \atop k}\right\} \;=\; k\left\{{n \atop k}\right\} + \left\{{n \atop k-1}\right\}, \qquad (27)$$

and tabulated the results for $n \le 11$. James Stirling, in his *Methodus Differentialis* (1730), had discovered the numbers $\left\{{n \atop k}\right\}$ in a purely algebraic context; thus Saka was the first person to realize their combinatorial significance.

An interesting algorithm for listing set partitions was subsequently devised by Toshiaki Honda (see exercise 24). Further details about genji-ko and its relation to the history of mathematics can be found in Japanese articles by Tamaki Yano, *Sugaku Seminar* **34**, 11 (Nov. 1995), 58–61; **34**, 12 (Dec. 1995), 56–60.

Set partitions remained virtually unknown in Europe until much later, except for three isolated incidents. First, George and/or Richard Puttenham published *The Arte of English Poesie* in 1589, and pages 70–72 of that book

contain diagrams similar to those of genji-ko. For example, the seven diagrams


(28)

were used to illustrate possible rhyme schemes for 5-line poems, "whereof some of them be harsher and unpleasaunter to the eare then other some be." But this visually appealing list was incomplete (see exercise 25).

Second, an unpublished manuscript of G. W. Leibniz from the late 1600s shows that he had tried to count the number of ways to partition $\{1,\ldots,n\}$ into three or four subsets, but with almost no success. He enumerated ${n \brace 2}$ by a very cumbersome method, which would not have led him to see readily that ${n \brace 2} = 2^{n-1} - 1$. He attempted to compute ${n \brace 3}$ and ${n \brace 4}$ only for $n \le 5$, and made several numerical slips leading to incorrect answers. [See E. Knobloch, *Studia Leibnitiana Supplementa* **11** (1973), 229–233; **16** (1976), 316–321.]

The third European appearance of set partitions had a completely different character. John Wallis devoted the third chapter of his *Discourse of Combinations* (1685) to questions about "aliquot parts," the proper divisors of numbers, and in particular he studied the set of all ways to factorize a given integer. This question is equivalent to the study of *multiset* partitions; for example, the factorizations of p^3q^2r are essentially the same as the partitions of $\{p,p,p,q,q,r\}$, when p, q, and r are prime numbers. Wallis devised an excellent algorithm for listing all factorizations of a given integer n, essentially anticipating Algorithm 7.2.1.5M (see exercise 28). But he didn't investigate the important special cases that arise when n is the power of a prime (equivalent to integer partitions) or when n is squarefree (equivalent to set partitions). Thus, although Wallis was able to solve the more general problem, its complexities paradoxically deflected him from discovering partition numbers, Bell numbers, or Stirling subset numbers, or from devising simple algorithms that would generate integer partitions or set partitions.

Integer partitions. Partitions of integers arrived on the scene even more slowly. Bishop Wibold (c. 965) knew the partitions of n into exactly three parts ≤ 6. So did Galileo, who wrote a memo about them (c. 1627) and also studied their frequency of occurrence as rolls of three dice. ["Sopra le scoperte de i dadi," in Galileo's *Opere*, Volume 8, 591–594; he listed partitions in decreasing lexicographic order.]

Mersenne listed the partitions of 9 into any number of parts, on page 130 of his *Traitez de la Voix et des Chants* (1636). With each partition $9 = a_1 + \cdots + a_k$ he also computed the multinomial coefficient $9!/(a_1! \ldots a_k!)$; as we've seen earlier, he was interested in counting various melodies, and he knew for example that there are $9!/(3!3!3!) = 1680$ melodies on the nine notes $\{a,a,a,b,b,b,c,c,c\}$. But he failed to mention the cases $8+1$ and $3+2+1+1+1+1$, probably because he hadn't listed the possibilities in any systematic way.

Leibniz considered two-part partitions in Problem 3 of his *Dissertatio de Arte Combinatoria* (1666), and his unpublished notes show that he subsequently spent considerable time trying to enumerate the partitions that have three or

more summands. He called them "discerptions," or (less frequently) "divulsions" — in Latin of course — or sometimes "sections" or "dispersions" or even "partitions." He was interested in them primarily because of their connection with the monomial symmetric functions $\sum x_{i_1}^{a_1} x_{i_2}^{a_2} \ldots$. But his many attempts led to almost total failure, except in the case of three summands, when he almost (but not quite) discovered the formula for $\left|{n \atop 3}\right|$ in exercise 7.2.1.4–31. For example, he carelessly counted only 21 partitions of 8, forgetting the case $2+2+2+1+1$; and he got only 26 for $p(9)$, after missing $3+2+2+2$, $3+2+2+1+1$, $2+2+2+1+1+1$, and $2+2+1+1+1+1+1$ — in spite of the fact that he was trying to list partitions systematically in decreasing lexicographic order. [See E. Knobloch, *Studia Leibnitiana Supplementa* **11** (1973), 91–258; **16** (1976), 255–337; *Historia Mathematica* **1** (1974), 409–430.]

Abraham de Moivre had the first real success with partitions, in his paper "A Method of Raising an infinite Multinomial to any given Power, or Extracting any given Root of the same" [*Philosophical Transactions* **19** (1697), 619–625 and Fig. 5]. He proved that the coefficient of z^{m+n} in $(az + bz^2 + cz^3 + \cdots)^m$ has one term for each partition of n; for example, the coefficient of z^{m+6} is

$$\begin{aligned}\tbinom{m}{6}a^{m-6}b^6 + 5\tbinom{m}{5}a^{m-5}b^4c + 4\tbinom{m}{4}a^{m-4}b^3d + 6\tbinom{m}{4}a^{m-4}b^2c^2 \\ + 3\tbinom{m}{3}a^{m-3}b^2e + 6\tbinom{m}{3}a^{m-3}bcd + 2\tbinom{m}{2}a^{m-2}bf + \tbinom{m}{3}a^{m-3}c^3 \\ + 2\tbinom{m}{2}a^{m-2}ce + \tbinom{m}{2}a^{m-2}d^2 + \tbinom{m}{1}a^{m-1}g. \qquad (29)\end{aligned}$$

If we set $a = 1$, the term with exponents $b^i c^j d^k e^l \ldots$ corresponds to the partition with i 1s, j 2s, k 3s, l 4s, etc. Thus, for example, when $n = 6$ he essentially presented the partitions in the order

$$111111,\ 11112,\ 1113,\ 1122,\ 114,\ 123,\ 15,\ 222,\ 24,\ 33,\ 6. \qquad (30)$$

He explained how to list the partitions recursively, as follows (but in different language related to his own notation): For $k = 1, 2, \ldots, n$, start with k and append the (previously listed) partitions of $n - k$ whose smallest part is $\geq k$.

[My solution] was ordered to be published in the Transactions,
not so much as a matter relating to Play,
but as containing some general Speculations
not unworthy to be considered by the Lovers of Truth.
— ABRAHAM DE MOIVRE (1717)

P. R. de Montmort tabulated all partitions of numbers ≤ 9 into ≤ 6 parts in his *Essay d'Analyse sur les Jeux de Hazard* (1708), in connection with dice problems. His partitions were listed in a different order from (30); for example,

$$111111,\ 21111,\ 2211,\ 222,\ 3111,\ 321,\ 33,\ 411,\ 42,\ 51,\ 6. \qquad (31)$$

He probably was unaware of de Moivre's prior work.

So far almost none of the authors we've been discussing actually described the procedures by which they generated combinatorial patterns. We can only infer their methods, or lack thereof, by studying the lists that they actually published. Furthermore, in rare cases such as de Moivre's paper where a tabulation

method *was* explicitly described, the author assumed that all patterns for the first cases 1, 2, ..., $n-1$ had been listed before it was time to tackle the case of order n. No method for generating patterns "on the fly," moving directly from one pattern to its successor without looking at auxiliary tables, was actually explained by any of the authors we have encountered, except for Kedāra and Nārāyaṇa. Today's computer programmers naturally prefer methods that are more direct and need little memory.

Roger Joseph Boscovich published the first direct algorithm for partition generation in *Giornale de' Letterati* (Rome, 1747), on pages 393–404 together with two foldout tables facing page 404. His method, which produces for $n = 6$ the respective outputs

$$111111,\ 11112,\ 1122,\ 222,\ 1113,\ 123,\ 33,\ 114,\ 24,\ 15,\ 6, \qquad (32)$$

generates partitions in precisely the reverse order from which they are visited by Algorithm 7.2.1.4P; and his method would indeed have been featured in Section 7.2.1.4, except for the fact that the reverse order turns out to be slightly easier and faster than the order that he had chosen.

Boscovich published sequels in *Giornale de' Letterati* (Rome, 1748), 12–27 and 84–99, extending his algorithm in two ways. First, he considered generating only partitions whose parts belong to a given set S, so that symbolic multinomials with sparse coefficients could be raised to the mth power. (He said that the gcd of all elements of S should be 1; in fact, however, his method could fail if $1 \notin S$.) Second, he introduced an algorithm for generating partitions of n into m parts, given m and n. Again he was unlucky: A slightly better way to do that task, Algorithm 7.2.1.4H, was found subsequently, diminishing his chances for fame.

Hindenburg's hype. The inventor of Algorithm 7.2.1.4H was Carl Friedrich Hindenburg, who also rediscovered Nārāyaṇa's Algorithm 7.2.1.2L, a winning technique for generating multiset permutations. Unfortunately, these small successes led him to believe that he had made revolutionary advances in mathematics — although he did condescend to remark that other people such as de Moivre, Euler, and Lambert had come close to making similar discoveries.

Hindenburg was a prototypical overachiever, extremely energetic if not inspired. He founded or cofounded Germany's first professional journals of mathematics (published 1786–1789 and 1794–1800), and contributed long articles to each. He served several times as academic dean at the University of Leipzig, where he was also the Rector in 1792. If he had been a better mathematician, German mathematics might well have flourished more in Leipzig than in Berlin or Göttingen.

But his first mathematical work, *Beschreibung einer ganz neuen Art, nach einem bekannten Gesetze fortgehende Zahlen durch Abzählen oder Abmessen bequem und sicher zu finden* (Leipzig: 1776), amply foreshadowed what was to come: His "ganz neue" (completely new) idea in that booklet was simply to give combinatorial significance to the digits of numbers written in decimal notation. Incredibly, he concluded his monograph with large foldout sheets that contained

a table of the numbers 0000 through 9999—followed by two other tables that listed the even numbers and odd numbers separately(!).

Hindenburg published letters from people who praised his work, and invited them to contribute to his journals. In 1796 he edited *Sammlung combinatorisch-analytischer Abhandlungen*, whose subtitle stated (in German) that de Moivre's multinomial theorem was "the most important proposition in all of mathematical analysis." About a dozen people joined forces to form what became known as Hindenburg's Combinatorial School, and they published thousands of pages filled with esoteric symbolism that must have impressed many nonmathematicians.

The work of this School was not completely trivial from the standpoint of computer science. For example, H. A. Rothe, who was Hindenburg's best student, noticed that there is a simple way to go from a Morse code sequence to its lexicographic successor or predecessor. Another student, J. C. Burkhardt, observed that Morse code sequences of length n could also be generated easily by first considering those with no dashes, then one dash, then two, etc. Their motivation was not to tabulate poetic meters of n beats, as it had been in India, but rather to list the terms of the continuant polynomials $K(x_1, x_2, \ldots, x_n)$, Eq. 4.5.3–(4). [See *Archiv für reine und angewandte Mathematik* **1** (1794), 154–194.] Furthermore, on page 53 of Hindenburg's 1796 *Sammlung* cited above, G. S. Klügel introduced a way to list all permutations that has subsequently become known as Ord-Smith's algorithm; see Eqs. (23)–(26) in Section 7.2.1.2.

Hindenburg believed that his methods deserved equal time with algebra, geometry, and calculus in the standard curriculum. But he and his disciples were combinatorialists who only made combinatorial lists. Burying themselves in formulas and formalisms, they rarely discovered any new mathematics of real interest. Eugen Netto has admirably summarized their work in M. Cantor's *Geschichte der Mathematik* **4** (1908), 201–219. "For a while they controlled the German market; however, most of what they dug up soon sank into a not-entirely-deserved oblivion."

The sad outcome was that combinatorial studies in general got a bad name. Gösta Mittag-Leffler, who assembled a magnificent library of mathematical literature about 100 years after Hindenburg's death, decided to place all such work on a special shelf marked "Dekadenter." And this category still persists in the library of Sweden's Institut Mittag-Leffler today, even as that institute attracts world-class combinatorial mathematicians whose research is anything but decadent.

Looking on the bright side, we may note that at least one good book did emerge from all of this activity. Andreas von Ettingshausen's *Die combinatorische Analysis* (Vienna: 1826) is noteworthy as the first text to discuss combinatorial generation methods in a perspicuous way. He discussed the general principles of lexicographic generation in §8, and applied them to construct good ways to list all permutations (§11), combinations (§30), and partitions (§41–§44).

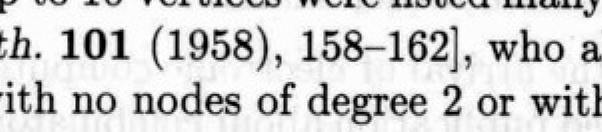
Where were the trees? We've now seen that lists of tuples, permutations, combinations, and partitions were compiled rather early in human history, by

interested and interesting researchers. Thus we've accounted for the evolution of the topics studied in Sections 7.2.1.1 through 7.2.1.5, and our story will be complete if we can trace the origins of tree generation, Section 7.2.1.6.

But the historical record of that topic before the advent of computers is virtually a blank page, with the exception of a few 19th-century papers by Arthur Cayley. Cayley's major work on trees, originally published in 1875 and reprinted on pages 427–460 of his *Collected Mathematical Papers*, Volume 4, was climaxed by a large foldout illustration that exhibited all the free trees with 9 or fewer unlabeled vertices. Earlier in that paper he had also illustrated the nine *oriented* trees with 5 vertices. The methods he used to produce those lists were quite complicated, completely different from Algorithm 7.2.1.6O and exercise 7.2.1.6–90. All free trees with up to 10 vertices were listed many years later by F. Harary and G. Prins [*Acta Math.* **101** (1958), 158–162], who also went up to $n = 12$ in the cases of free trees with no nodes of degree 2 or with no symmetries.

The trees most dearly beloved by computer scientists—binary trees or the equivalent ordered forests or nested parentheses—are however strangely absent from the literature. We saw in Section 2.3.4.5 that many mathematicians of the 1700s and 1800s had learned how to count binary trees, and we also know that the Catalan numbers C_n enumerate dozens of different kinds of combinatorial objects. Yet nobody seems to have published an actual *list* of the $C_4 = 14$ objects of order 4 in *any* of these guises, much less the $C_5 = 42$ objects of order 5, before 1950. (Except indirectly: The 42 genji-ko diagrams in (25) that have no intersecting lines turn out to be equivalent to the 5-node binary trees and forests. But this fact was not learned until the 20th century.)

There are a few isolated instances where authors of yore did prepare lists of $C_3 = 5$ Catalan-related objects. Cayley, again, was first; he illustrated the binary trees with 3 internal nodes and 4 leaves as follows in *Philosophical Magazine* **18** (1859), 374–378:


(33)

(That same paper also illustrated another species of tree, equivalent to so-called weak orderings.) Then, in 1901, E. Netto listed the five ways to insert parentheses into the expression '$a + b + c + d$':

$$(a+b)+(c+d),\ [(a+b)+c]+d,\ [a+(b+c)]+d,\ a+[(b+c)+d],\ a+[b+(c+d)]. \quad (34)$$

[*Lehrbuch der Combinatorik*, §122.] And the five permutations of $\{+1, +1, +1, -1, -1, -1\}$ whose partial sums are nonnegative were listed in the following way by Paul Erdős and Irving Kaplansky [*Scripta Math.* **12** (1946), 73–75]:

$$1{+}1{+}1{-}1{-}1{-}1,\quad 1{+}1{-}1{+}1{-}1{-}1,\quad 1{+}1{-}1{-}1{+}1{-}1,\quad 1{-}1{+}1{+}1{-}1{-}1,\quad 1{-}1{+}1{-}1{+}1{-}1. \quad (35)$$

Even though only five objects are involved, we can see that the orderings in (33) and (34) were basically catch-as-catch-can; only (35), which matches Algorithm 7.2.1.6P, was systematic and lexicographic.

We should also note briefly the work of Walther von Dyck, since many recent papers use the term "Dyck words" to refer to strings of nested parentheses. Dyck was an educator known for co-founding the Deutsches Museum in Munich, among other things. He wrote two pioneering papers about the theory of free groups [*Math. Annalen* **20** (1882), 1–44; **22** (1883), 70–108]. Yet the so-called Dyck words have at best a tenuous connection to his actual research: He studied the words on $\{x_1, x_1^{-1}, \ldots, x_k, x_k^{-1}\}$ that reduce to the empty string after repeatedly erasing adjacent letter-pairs of the forms $x_i x_i^{-1}$ or $x_i^{-1} x_i$; the connection with parentheses and trees arises only when we limit erasures to the first case, $x_i x_i^{-1}$.

Thus we may conclude that, although an explosion of interest in binary trees and their cousins occurred after 1950, such trees represent the only aspect of our story whose historical roots are rather shallow.

After 1950. Of course the arrival of electronic computers changed everything. The first computer-oriented publication about combinatorial generation methods was a note by C. B. Tompkins, "Machine attacks on problems whose variables are permutations" [*Proc. Symp. Applied Math.* **6** (1956), 202–205]. Thousands more were destined to follow.

Several articles by D. H. Lehmer, especially his "Teaching combinatorial tricks to a computer" in *Proc. Symp. Applied Math.* **10** (1960), 179–193, proved to be extremely influential in the early days. [See also *Proc. 1957 Canadian Math. Congress* (1959), 160–173; *Proc. IBM Scientific Computing Symposium on Combinatorial Problems* (1964), 23–30; and Chapter 1 of *Applied Combinatorial Mathematics*, edited by E. F. Beckenbach (Wiley, 1964), 5–31.] Lehmer represented an important link to previous generations. For example, Stanford's library records show that he had checked out Netto's *Lehrbuch der Combinatorik* in January of 1932.

The main publications relevant to particular algorithms that we've studied have already been cited in previous sections, so there is no need to repeat them here. But textbooks and monographs that first put pieces of the subject together in a coherent framework were also of great importance. Three books, in particular, were especially noteworthy with respect to establishing general principles:

- *Elements of Combinatorial Computing* by Mark B. Wells (Pergamon Press, 1971), especially Chapter 5.
- *Combinatorial Algorithms* by Albert Nijenhuis and Herbert S. Wilf (Academic Press, 1975). A second edition was published in 1978, containing additional material, and Wilf subsequently wrote *Combinatorial Algorithms: An Update* (Philadelphia: SIAM, 1989).
- *Combinatorial Algorithms: Theory and Practice* by Edward M. Reingold, Jurg Nievergelt, and Narsingh Deo (Prentice–Hall, 1977), especially the material in Chapter 5.

Robert Sedgewick compiled the first extensive survey of permutation generation methods in *Computing Surveys* **9** (1977), 137–164, 314. Carla Savage's survey article about Gray codes in *SIAM Review* **39** (1997), 605–629, was another milestone.

We noted above that algorithms to generate Catalan-counted objects were not invented until computer programmers developed an appetite for them. The first such algorithms to be published were not cited in Section 7.2.1.6 because they have been superseded by better techniques; but it is appropriate to list them here. First, H. I. Scoins gave two recursive algorithms for ordered tree generation, in the same paper we have cited with respect to the generation of *oriented* trees [*Machine Intelligence* **3** (1968), 43–60]. His algorithms dealt with binary trees represented as bit strings that were essentially equivalent to Polish prefix notation or to nested parentheses. Then Mark Wells, in Section 5.5.4 of his book cited above, generated binary trees by representing them as noncrossing set partitions. And Gary Knott [*CACM* **20** (1977), 113–115] gave recursive ranking and unranking algorithms for binary trees, representing them via the inorder-to-preorder permutations $q_1 \ldots q_n$ of Table 7.2.1.6–3.

Algorithms to generate all spanning trees of a given graph have been published by numerous authors ever since the 1950s, motivated originally by the study of electrical networks. Among the earliest such papers were works of N. Nakagawa, *IRE Trans.* **CT-5** (1958), 122–127; W. Mayeda, *IRE Trans.* **CT-6** (1959), 136–137, 394; H. Watanabe, *IRE Trans.* **CT-7** (1960), 296–302; S. Hakimi, *J. Franklin Institute* **272** (1961), 347–359.

A recent introduction to the entire subject can be found in Chapters 2 and 3 of *Combinatorial Algorithms: Generation, Enumeration, and Search* by Donald L. Kreher and Douglas R. Stinson (CRC Press, 1999).

Frank Ruskey is preparing a book entitled *Combinatorial Generation* that will contain a thorough treatment and a comprehensive bibliography. He has made working drafts of several chapters available on the Internet.

EXERCISES

Many of the exercises below ask a modern reader to find and/or to correct errors in the literature of bygone days. The point is not to gloat over how smart we are in the 21st century; the point is rather to understand that even the pioneers of a subject can stumble. One good way to learn that a set of ideas is not really as simple as it might seem to today's computer scientists and mathematicians is to observe that some of the world's leading thinkers had to struggle with the concepts when they were new.

1. [*15*] Does the notion of "computing" arise in the *I Ching*?

▸ **2.** [*M30*] (*The genetic code.*) DNA molecules are strings of "nucleotides" on the 4-letter alphabet {T, C, A, G}, and most protein molecules are strings of "amino acids" on the 20-letter alphabet {A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y}. Three consecutive nucleotides xyz form a "codon," and a strand $x_1y_1z_1x_2y_2z_2\ldots$ of DNA specifies the protein $f(x_1,y_1,z_1)f(x_2,y_2,z_2)\ldots$, where $f(x,y,z)$ is the element in row z and column y of matrix x in the array

$$\begin{pmatrix} F & S & Y & C \\ F & S & Y & C \\ L & S & - & - \\ L & S & - & W \end{pmatrix} \begin{pmatrix} L & P & H & R \\ L & P & H & R \\ L & P & Q & R \\ L & P & Q & R \end{pmatrix} \begin{pmatrix} I & T & N & S \\ I & T & N & S \\ I & T & K & R \\ M & T & K & R \end{pmatrix} \begin{pmatrix} V & A & D & G \\ V & A & D & G \\ V & A & E & G \\ V & A & E & G \end{pmatrix}.$$

(Here (T, C, A, G) = (1, 2, 3, 4); for example, $f(\mathtt{CAT})$ is the element in row 1 and column 3 of matrix 2, namely H.) Encoding proceeds until a codon leads to the stopper '–'.

a) Show that there is a simple way to map each codon into a hexagram of the *I Ching*, with the property that the 21 possible outcomes $\{A, C, D, \ldots, W, Y, -\}$ correspond to 21 *consecutive* hexagrams of the King Wen ordering (1).

b) Is that a sensational discovery?

3. [*20*] What is the millionth meter that has 30 beats, in colex ordering analogous to (2)? What is the rank of ⌣⌣⌣—⌣——⌣⌣⌣⌣——⌣⌣⌣⌣⌣⌣⌣—⌣—?

4. [*19*] Analyze the imperfections of Donnolo's list of permutations in Table 1.

5. [*16*] What's wrong with Kircher's list of five-note permutations in (7)?

6. [*25*] Mersenne published a table of the first 64 factorials on pages 108–110 of his *Traitez de la Voix et des Chants* (1636). His value for 64! was $\approx 2.2 \times 10^{89}$; but it should have been $\approx 1.3 \times 10^{89}$. Find a copy of his book and try to figure out where he erred.

7. [*20*] What permutations of $\{1, 2, 3, 4, 5\}$ are "alive" and "dead" according to Seki's rules (8) and (9)?

▸ **8.** [*M27*] Make a patch to (9) so that Seki's procedure will be correct.

9. [*15*] From (11), deduce the Arabic way to write the Arabic numerals $(0, 1, \ldots, 9)$.

▸ **10.** [*HM27*] In Ludus Clericalis, what is the expected number of times the three dice are rolled before all possible virtues are acquired?

11. [*21*] Decipher Llull's vertical table at the right of Fig. 45. What 20 combinatorial objects does it represent? *Hint:* Don't be misled by typographic errors.

12. [*M20*] Relate Schillinger's universal cycle (13) to the universal cycle of Poinsot in exercise 7.2.1.3–106.

13. [*21*] What should van Schooten have written, instead of (17)? Give also the corresponding tableau for combinations of the multiset $\{a, a, a, b, b, c\}$.

▸ **14.** [*20*] Complete the following sequence, from §95 of Izquierdo's *De Combinatione*:

ABC ABD ABE ACD ACE ACB ADE ADB ADC AEB . . . .

15. [*15*] If all n-combinations of $\{1, \ldots, m\}$ with repetition are listed in lexicographic order, how many of them begin with the number j?

16. [*20*] (Nārāyaṇa Paṇḍita, 1356.) Design an algorithm to generate all compositions of n into parts $\le q$, namely all ordered partitions $n = a_1 + \cdots + a_t$, where $1 \le a_j \le q$ for $1 \le j \le t$ and t is arbitrary. Illustrate your method when $n = 7$ and $q = 3$.

17. [*HM27*] Analyze the algorithm of exercise 15.

18. [*10*] Trick question: Leibniz published his *Dissertatio de Arte Combinatoria* in 1666. Why was that a particularly auspicious year, permutationwise?

19. [*17*] In which of Puteanus's verses (20) is 'tibi' treated as ⌣— instead of ⌣⌣?

20. [*M25*] To commemorate the visit of three illustrious noblemen to Dresden in 1617, a poet published 1617 permutations of the hexameter verse

Dant tria jam Dresdæ, ceu sol dat, lumina lucem.

"Three give now to Dresden, as the sun gives, lights to light." [Gregor Kleppis, *Proteus Poeticus* (Leipzig: 1617).] How many permutations of those words would actually scan properly? *Hint:* The verse has dactyls in the first and fifth feet, spondees elsewhere.

21. [*HM30*] Let $f(p,q,r;s,t)$ be the number of ways to make (o^p, o^q, o^r) by concatenating the strings $\{s\cdot o, t\cdot oo\}$, when $p+q+r=s+2t$. For example, $f(2,3,2;3,2)=5$ because the five ways are

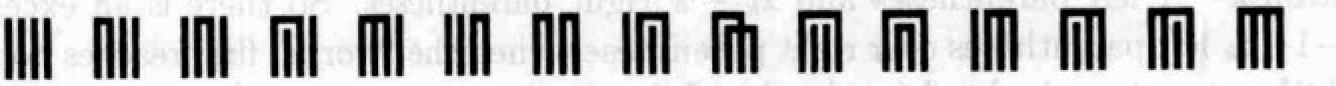
$$(o|o, o|oo, oo),\quad (o|o, oo|o, oo),\quad (oo, o|o|o, oo),\quad (oo, o|oo, o|o),\quad (oo, oo|o, o|o).$$

a) Show that $f(p,q,r;s,t) = [u^p v^q w^r z^s]\, 1/((1-zu-u^2)(1-zv-v^2)(1-zw-w^2))$.
b) Use the function f to enumerate the scannable permutations of (19), subject to the additional condition that the fifth foot doesn't begin in the middle of a word.
c) Now enumerate the remaining cases.

▸ **22.** [*M40*] Look up the original discussions of the tot-tibi problem that were published by Prestet, Wallis, Whitworth, and Hartley. What errors did they make?

23. [*20*] What order of the 52 genji-ko diagrams corresponds to Algorithm 7.2.1.5H?

▸ **24.** [*23*] Early in the 1800s, Toshiaki Honda gave a recursive rule for generating all partitions of $\{1,\ldots,n\}$. His algorithm produced them in the following order when $n=4$:



Can you guess the corresponding order for $n=5$? *Hint:* See (26).

25. [*15*] The 16th-century author of *The Arte of English Poesie* was interested only in rhyme schemes that are "complete" in the sense of exercise 7.2.1.5–35; in other words, every line should rhyme with at least one other. Furthermore, the scheme should be "indecomposable" in the sense of exercise 7.2.1.2–100: A partition like $12|345$ decomposes into a 2-line poem followed by a 3-line poem. And the scheme shouldn't consist trivially of lines that all rhyme with each other. Under these conditions, is (28) a complete list of 5-line rhyme schemes?

▸ **26.** [*HM25*] How many n-line rhyme schemes satisfy the constraints of exercise 25?

▸ **27.** [*HM31*] The set partition $14|25|36$ can be represented by a genji-ko diagram such as ; but every such diagram for this partition must have at least three places where lines cross, and crossings are sometimes considered undesirable. How many partitions of $\{1,\ldots,n\}$ have a genji-ko diagram in which the lines cross at most once?

▸ **28.** [*25*] Let a, b, and c be prime numbers. John Wallis listed all possible factorizations of a^3b^2c as follows: $cbbaaa$, $cbbaa\cdot a$, $bcaaa\cdot b$, $bbaaa\cdot c$, $cbba\cdot aa$, $cbba\cdot a\cdot a$, $cbaa\cdot ba$, $cbaa\cdot b\cdot a$, $bbaa\cdot ca$, $bbaa\cdot c\cdot a$, $caaa\cdot bb$, $caaa\cdot b\cdot b$, $baaa\cdot cb$, $baaa\cdot c\cdot b$, $cbb\cdot aaa$, $cbb\cdot aa\cdot a$, $cbb\cdot a\cdot a\cdot a$, $cba\cdot baa$, $cba\cdot ba\cdot a$, $cba\cdot aa\cdot b$, $cba\cdot b\cdot a\cdot a$, $bba\cdot caa$, $bba\cdot ca\cdot a$, $bba\cdot aa\cdot c$, $bba\cdot c\cdot a\cdot a$, $caa\cdot bb\cdot a$, $caa\cdot ba\cdot b$, $caa\cdot b\cdot b\cdot a$, $baa\cdot cb\cdot a$, $baa\cdot ca\cdot b$, $baa\cdot ba\cdot c$, $baa\cdot c\cdot b\cdot a$, $aaa\cdot cb\cdot b$, $aaa\cdot bb\cdot c$, $aaa\cdot c\cdot b\cdot b$, $cb\cdot ba\cdot aa$, $cb\cdot ba\cdot a\cdot a$, $cb\cdot aa\cdot b\cdot a$, $cb\cdot b\cdot a\cdot a\cdot a$, $bb\cdot ca\cdot aa$, $bb\cdot ca\cdot a\cdot a$, $bb\cdot aa\cdot c\cdot a$, $bb\cdot c\cdot a\cdot a\cdot a$, $ca\cdot ba\cdot ba$, $ca\cdot ba\cdot b\cdot a$, $ca\cdot aa\cdot b\cdot b$, $ca\cdot b\cdot b\cdot a\cdot a$, $ba\cdot ba\cdot c\cdot a$, $ba\cdot aa\cdot c\cdot b$, $ba\cdot c\cdot b\cdot a\cdot a$, $aa\cdot c\cdot b\cdot b\cdot a$, $c\cdot b\cdot b\cdot a\cdot a\cdot a$. What algorithm did he use to generate them in this order?

▸ **29.** [*24*] In what order would Wallis have generated all factorizations of the number $abcde = 5\cdot 7\cdot 11\cdot 13\cdot 17$? Give your answer as a sequence of genji-ko diagrams.

30. [*M20*] What is the coefficient of $a_1^{i_1}a_2^{i_2}\ldots z^{m+n}$ in $(a_0z + a_1z^2 + a_2z^3 + \cdots)^m$? (See (29).)

31. [*20*] Compare de Moivre's and de Montmort's orders for partitions, (30) and (31), with Algorithm 7.2.1.4P.

32. [*21*] (R. J. Boscovich, 1748.) List all partitions of 20 for which all parts are 1, 7, or 10. Also design an algorithm that lists all such partitions of any given integer $n>0$.

SECTION 7.2.1.6

1. It could "see" a left parenthesis at the left of every internal node and a right parenthesis at the bottom of every internal node. Alternatively, it could associate right parentheses with the *external* nodes that it encounters — except for the very last □; see exercise 20.

2. **Z1.** [Initialize.] Set $z_k \leftarrow 2k-1$ for $0 \le k \le n$. (Assume that $n \ge 2$.)

Z2. [Visit.] Visit the tree-combination $z_1 z_2 \ldots z_n$.

Z3. [Easy case?] If $z_{n-1} < z_n - 1$, set $z_n \leftarrow z_n - 1$ and return to Z2.

Z4. [Find j.] Set $j \leftarrow n-1$ and $z_n \leftarrow 2n-1$. While $z_{j-1} = z_j - 1$, set $z_j \leftarrow 2j-1$ and $j \leftarrow j-1$.

Z5. [Decrease z_j]. Terminate the algorithm if $j = 1$. Otherwise set $z_j \leftarrow z_j - 1$ and go back to Z2. ▮

3. Label the nodes of the forest in preorder. The first $z_k - 1$ elements of $a_1 \ldots a_{2n}$ contain $k-1$ left parentheses and $z_k - k$ right parentheses. So there is an excess of $2k-1-z_k$ left parentheses over right parentheses when the "worm" first reaches node k; and $2k-1-z_k$ is the level (or depth) of that node.

Let $q_1 \ldots q_n$ be the inverse of $p_1 \ldots p_n$, so that node k is the q_kth node in postorder. Since k occurs to the left of j in $p_1 \ldots p_n$ if and only if $q_k < q_j$, we see that c_k is the number of nodes j that precede k in preorder but follow it in postorder, namely the number of ancestors of k; again, this is the level of k.

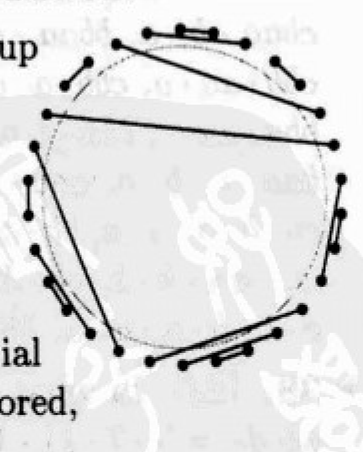
Alternative proof: We can also show that both sequences $z_1 \ldots z_n$ and $c_1 \ldots c_n$ have essentially the same recursive structure as (5): $Z_{pq} = (Z_{p(q-1)} + 1^p),\ 1(Z_{(p-1)q} + 1^{p-1})$ when $0 \le p \le q$; and $C_{pq} = C_{p(q-1)},\ (q-p)C_{(p-1)q}$. (Consider the mate of the last, next-to-last, etc., left parenthesis.)

Incidentally, the formula '$c_{k+1} + d_k = c_k + 1$' is equivalent to (11).

4. Almost true; but $d_1 \ldots d_n$ and $z_1 \ldots z_n$ occur in *decreasing* order, while $p_1 \ldots p_n$ and $c_1 \ldots c_n$ are increasing. (This lexicographic property for a sequence of permutations $p_1 \ldots p_n$ is not automatically inherited from lexicographic order of the corresponding inversion tables $c_1 \ldots c_n$; but the result does hold for this particular class of $p_1 \ldots p_n$.)

5. $d_1 \ldots d_{15} = 0\,2\,0\,0\,2\,0\,0\,1\,0\,3\,2\,0\,1\,0\,4$; $z_1 \ldots z_{15} = 1\,2\,5\,6\,7\,10\,11\,12\,14\,15\,19\,22\,23\,25\,26$; $p_1 \ldots p_{15} = 2\,1\,5\,4\,8\,10\,9\,7\,11\,6\,13\,15\,14\,12\,3$; $c_1 \ldots c_{15} = 0\,1\,0\,1\,2\,1\,2\,3\,3\,4\,2\,1\,2\,2\,3$.

6. Match up the parentheses as usual; then simply curl the string up and around until a_{2n} becomes adjacent to a_1, and notice that the distinction between left and right parentheses can be reconstructed from the context. Letting a_1 correspond to the bottom of the circle, as in Table 1, yields the diagram shown. [A. Errera, *Mémoires de la Classe Sci. 8°, Acad. Royale de Belgique* (2) **11**, 6 (1931), 26 pp.]



7. (a) It equals))()…(); setting $a_1 \leftarrow$ '(' will restore the initial string. (b) The initial binary tree (from step B1) will have been restored, except that $l_n = n+1$.

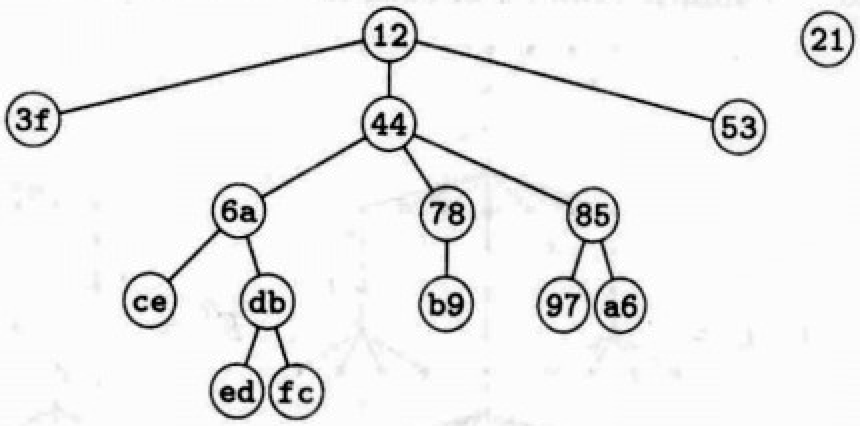
8. $l_1 \ldots l_{15} = 2\,0\,4\,5\,0\,7\,8\,0\,10\,0\,0\,13\,0\,15\,0$; $r_1 \ldots r_{15} = 3\,0\,0\,6\,0\,12\,11\,9\,0\,0\,0\,0\,14\,0\,0$; $e_1 \ldots e_{15} = 1\,0\,3\,1\,0\,2\,2\,0\,1\,0\,0\,2\,0\,1\,0$; $s_1 \ldots s_{15} = 1\,0\,12\,1\,0\,5\,3\,0\,1\,0\,0\,3\,0\,1\,0$.

9. Node j is an ancestor of node k if and only if $s_j + j \ge k$. (As a consequence, we have $c_1 + \cdots + c_n = s_1 + \cdots + s_n$.)

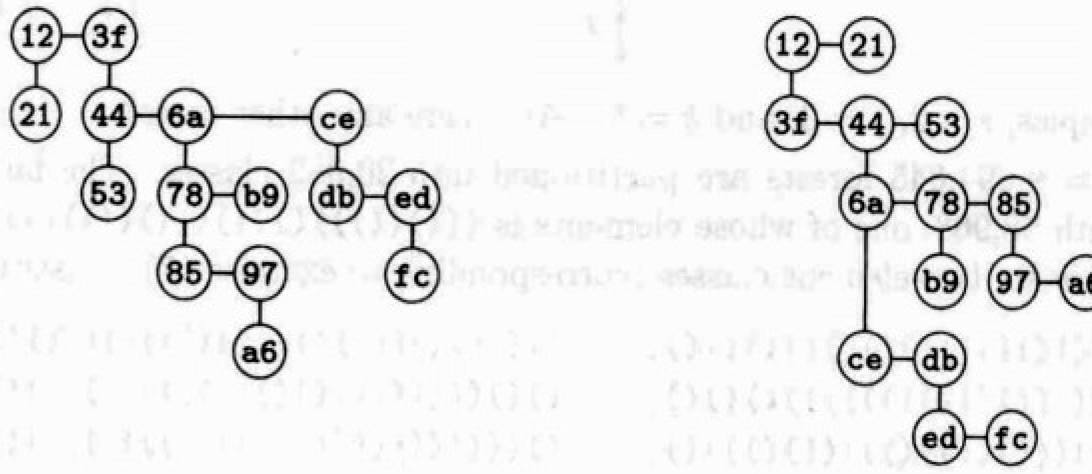
10. If j is the index z_k of the kth left parenthesis, we have $w_j = c_k + 1$ and $w_{j'} = c_k$, where j' is the index of the matching right parenthesis.

11. Swap left and right parentheses in $a_{2n} \ldots a_1$ to get the mirror image of $a_1 \ldots a_{2n}$.

12. The mirror reflection of (4) corresponds to the forest



but the significance of transposition is clearer, forest-wise, if we draw right-sibling and left-child links horizontally and vertically, then do a matrix-like transposition:



13. (a) By induction on the number of nodes, we have $\text{preorder}(F^R) = \text{postorder}(F)^R$ and $\text{postorder}(F^R) = \text{preorder}(F)^R$.

(b) Let F correspond to the binary tree B; then $\text{preorder}(F) = \text{preorder}(B)$ and $\text{postorder}(F) = \text{inorder}(B)$, as noted after 2.3.2–(6). Therefore $\text{preorder}(F^T) = \text{preorder}(B^R) = \text{postorder}(B)^R$ has no simple relationship to either $\text{preorder}(F)$ or $\text{postorder}(F)$. But $\text{postorder}(F^T) = \text{inorder}(B^R) = \text{inorder}(B)^R = \text{postorder}(F)^R$.

14. According to answer 13, $\text{postorder}(F^{RT}) = \text{preorder}(F) = \text{preorder}(B)$ when F corresponds naturally to B; and $\text{postorder}(F^{TR}) = \text{preorder}(F^T)^R = \text{postorder}(B)$. Therefore the equation $F^{RT} = F^{TR}$ holds if and only if F has at most one node.

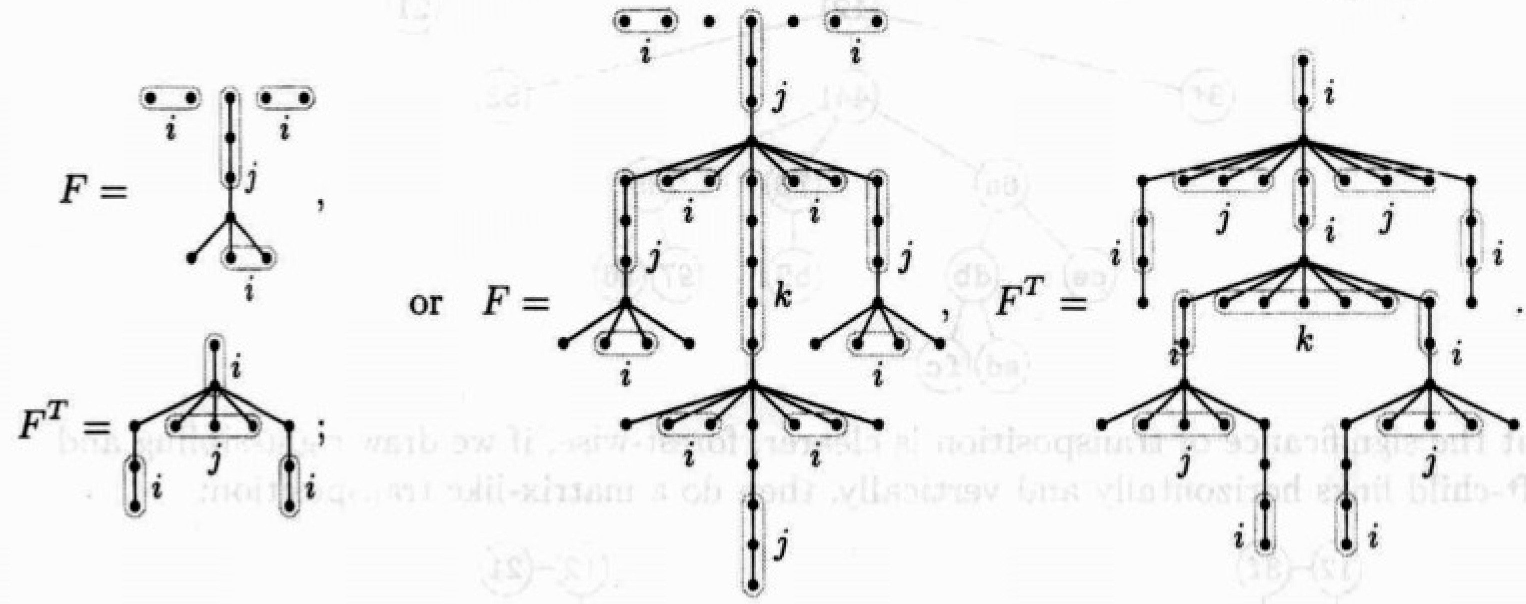
15. If F^R corresponds naturally to the binary tree B', the root of B' is the root of F's rightmost tree. The left link of node x in B' is to the leftmost child of x in F^R, which is the rightmost child of x in F; similarly, the right link is to x's left sibling in F.

Note: Since B corresponds naturally to F^{RT}, answer 13 tells us that $\text{inorder}(B) = \text{postorder}(F^{RT}) = \text{postorder}(F^R)^R = \text{preorder}(F)$.

16. The forest $F \mid G$ is obtained by placing the trees of F below the first node of G in postorder. Associativity follows because $F|(G|H) = (H^T G^T F^T)^T = (F|G)|H$. Notice, incidentally, that $\text{postorder}(F \mid G) = \text{postorder}(F)\,\text{postorder}(G)$, and that $F \mid (GH) = (F \mid G)H$ when G is nonnull.

17. Any nonnull forest can be written $F = (G|\cdot)H$, where $\cdot$ denotes the 1-node forest; then $F^R = H^R(G^R \mid \cdot)$ and $F^T = (H^T \mid \cdot)G^T$. In particular we cannot have $F^R = F^T$ unless H is the null forest Λ, since the first tree of H^R can't be $H^T \mid \cdot$; and G must then also be Λ. Furthermore $F = F^T$ if and only if $G = H^T$. In that case we cannot also have $F^R = F^{RT}$ unless $G = \Lambda$; the first tree of G^{TR} would otherwise have more nodes than G itself.

It appears to be true that we cannot have $F^{RT} = F^{TR}$ unless $F = F^R$. Under that assumption, $F^{RT} = F^{TR}$ if and only if F and F^T are both self-conjugate. David Callan has discovered two infinite families of such forests, with parameters $i, j, k \geq 0$:



(In these examples, $i = 2$, $j = 3$, and $k = 5$.) Are there any other cases?

18. The C_{15} = 9,694,845 forests are partitioned into 20,982 classes. The largest is a cycle of length 58,968, one of whose elements is ((()(()))())()((()(())())())(). The shortest are six two-element classes (corresponding to exercise 17), consisting of

()()()()()()()()()()()()()()(), ()()()()((()()()()()))()()()(),
()()()((((()()()()))))()()(), ()()((((((((()()()))))))))()(),
()((()())((()(())()))(()()))(), ()(((((((((((()())))))))))))(),

and their transposes. The somewhat strange strings (((((((())))))))()()()()()()(), ()()()()()()()(((((((()))))))), and (((((((()()()()()()()()))))))) each have wedge-shaped binary trees and form a unique class of size 3. The path that runs from ()((()(()()))(())((()())()))() to ((()())(()())(())(()())(()())) has 3120 elements, one of which is (2). According to the conjecture in answer 19, the shortest possible cycle has length 6; when $n = 15$ there are 66 such cycles. (The next-shortest cycle, which is unique, has length 10 and includes ()(()()())()((((())()))((())))).)

19. The transformation from F_j to F_{j+1} by Algorithm P can be paraphrased as follows: "Find the last node in preorder, say x, that has a left sibling, say y. Remove x from its family and make it the new rightmost child of y. And if $x < n$, change all of x's descendants $x + 1, \ldots, n$ into trivial one-node trees."

The transformation that takes F_j^R into F_{j+1}^R can therefore be stated as follows, if we recall that the kth node of F_j in preorder is the kth-from-last node of F_j^R in postorder: "Find the first node in preorder, say x, that has a right sibling, say y. Remove x from its family and make it the new leftmost child of y. And if $x < n$, change all of x's descendants $x + 1, \ldots, n$ into trivial one-node trees."

Similarly, we can paraphrase the transformation from G_j to G_{j+1} that is specified by Algorithm B: "Find j, the root of the leftmost nontrivial tree; then find k, its rightmost child. Remove k and its descendants from j's family, and insert them between j and j's right sibling. Finally, if $j > 1$, make j and its siblings all children of $j - 1$, and $j - 1$ a child of $j - 2$, etc."

When this transformation changes the left-sibling/right-child representation from G_j^{RT} to G_{j+1}^{RT} (see exercise 15), it turns out to be identical to the transformation that takes F_j^R to F_{j+1}^R in the left-child/right-sibling representation. Therefore $G_j^{RT} = F_j^R$, because this identity clearly holds when $j = 1$.

(It follows that the sequence of tables $e_1 \ldots e_{n-1}$ for the binary trees generated by Algorithm B is exactly the sequence of tables $d_{n-1} \ldots d_1$ for the parenthesis strings generated by Algorithm P; this phenomenon is illustrated in Tables 1 and 2.)

The forest F^{RTR} is called the *dual* of F; see exercise 26(f). Several symmetries between lists of forests have been explored by M. C. Er, *Comp. J.* **32** (1989), 76–85.

20. (a) This assertion, which generalizes Lemma 2.3.1P, is readily proved by induction.

(b) The following procedure is, in fact, almost identical to Algorithm P:

T1. [Initialize.] Set $b_{3k-2} \leftarrow 3$ and $b_{3k-1} \leftarrow b_{3k} \leftarrow 0$ for $1 \le k \le n$; also set $b_0 \leftarrow b_N \leftarrow 0$ and $m \leftarrow N-3$, where $N = 3n+1$.

T2. [Visit.] Visit $b_1 \ldots b_N$. (Now $b_m = 3$ and $b_{m+1} \ldots b_N = 0 \ldots 0$.)

T3. [Easy case?] Set $b_m \leftarrow 0$. If $b_{m-1} = 0$, set $b_{m-1} \leftarrow 3$, $m \leftarrow m-1$, and go to T2.

T4. [Find j.] Set $j \leftarrow m-1$ and $k \leftarrow N-3$. While $b_j = 3$, set $b_j \leftarrow 0$, $b_k \leftarrow 3$, $j \leftarrow j-1$, and $k \leftarrow k-3$.

T5. [Increase b_j.] Terminate the algorithm if $j = 0$. Otherwise set $b_j \leftarrow 3$, $m \leftarrow N-3$, and return to T2. ▌

[See S. Zaks, *Theoretical Comp. Sci.* **10** (1980), 63–82. In that article, Zaks pointed out that it is even easier to generate the sequence $z_1 \ldots z_n$ of indices j such that $b_j = 3$, using an algorithm virtually identical to the answer to exercise 2, because a valid ternary tree combination $z_1 \ldots z_n$ is characterized by the inequalities $z_{k-1} < z_k \le 3k-2$.]

21. For this problem we can essentially combine Algorithm P with Algorithm 7.2.1.2L. We shall assume for convenience that $n_t > 0$ and $n_1 + \cdots + n_t > 1$.

G1. [Initialize.] Set $l \leftarrow N$. Then for $j = t, \ldots, 2, 1$ (in this order), do the following operations n_j times: Set $b_{l-j} \leftarrow j$, $b_{l-j+1} \leftarrow \cdots \leftarrow b_{l-1} \leftarrow 0$, and $l \leftarrow l-j$. Finally set $b_0 \leftarrow b_N \leftarrow c_0 \leftarrow 0$ and $m \leftarrow N-t$.

G2. [Visit.] Visit $b_1 \ldots b_N$. (At this point $b_m > 0$ and $b_{m+1} = \cdots = b_N = 0$.)

G3. [Easy case?] If $b_{m-1} = 0$, set $b_{m-1} \leftarrow b_m$, $b_m \leftarrow 0$, $m \leftarrow m-1$, and return to G2.

G4. [Find j.] Set $c_1 \leftarrow b_m$, $b_m \leftarrow 0$, $j \leftarrow m-1$, and $k \leftarrow 1$. While $b_j \ge c_k$, set $k \leftarrow k+1$, $c_k \leftarrow b_j$, $b_j \leftarrow 0$, and $j \leftarrow j-1$.

G5. [Increase b_j.] If $b_j > 0$, find the smallest $l \ge 1$ such that $b_j < c_l$, and interchange $b_j \leftrightarrow c_l$. Otherwise, if $j > 0$, set $b_j \leftarrow c_1$ and $c_1 \leftarrow 0$. Otherwise terminate.

G6. [Reverse and spread out.] Set $j \leftarrow k$ and $l \leftarrow N$. While $c_j > 0$, set $b_{l-c_j} \leftarrow c_j$, $l \leftarrow l - c_j$, and $j \leftarrow j-1$. Then set $m \leftarrow N - c_k$ and go back to G2. ▌

This algorithm assumes that $N > n_1 + 2n_2 + \cdots + tn_t$. [See *SICOMP* **8** (1979), 73–81.]

22. Note first that d_1 can be increased if and only if $r_1 = 0$ in the linked representation. Otherwise the successor of $d_1 \ldots d_{n-1}$ is obtained by finding the smallest j with $d_j > 0$ and setting $d_j \leftarrow 0$, $d_{j+1} \leftarrow d_{j+1} + 1$. We may assume that $n > 2$.

K1. [Initialize.] Set $l_k \leftarrow k+1$ and $r_k \leftarrow 0$ for $1 \le k < n$; also set $l_n \leftarrow r_n \leftarrow 0$.

K2. [Visit.] Visit the binary tree represented by $l_1 l_2 \ldots l_n$ and $r_1 r_2 \ldots r_n$.

K3. [Easy cases?] Set $y \leftarrow r_1$. If $y = 0$, set $r_1 \leftarrow 2$, $l_1 \leftarrow 0$, and return to K2. Otherwise if $l_1 = 0$, set $l_1 \leftarrow 2$, $r_1 \leftarrow r_2$, $r_2 \leftarrow l_2$, $l_2 \leftarrow 0$, and return to K2. Otherwise set $j \leftarrow 2$ and $k \leftarrow 1$.

K4. [Find j and k.] If $r_j > 0$, set $k \leftarrow j$ and $y \leftarrow r_j$. Then if $j \ne y-1$, set $j \leftarrow j+1$ and repeat this step.

K5. [Shuffle subtrees.] Set $l_j \leftarrow y$, $r_j \leftarrow r_y$, $r_y \leftarrow l_y$, and $l_y \leftarrow 0$. If $j = k$, go to K2.

K6. [Shift subtrees.] Terminate if $y = n$. Otherwise, while $k > 1$, set $k \leftarrow k-1$, $j \leftarrow j-1$, and $r_j \leftarrow r_k$. Then while $j > 1$, set $j \leftarrow j-1$ and $r_j \leftarrow 0$. Return to K2. ▮

(See the analysis in exercise 45. Korsh [*Comp. J.* **48** (2005), 488–497] has shown that this algorithm can be extended in an interesting way to t-ary trees; and he has also found an efficient t-ary generalization of Algorithm B.)

23. (a) Since z_n begins at $2n-1$ and goes back and forth C_{n-1} times, it ends at $2n-1-(C_{n-1} \bmod 2)$, when $n > 1$. Furthermore the final value of z_j is constant for all $n \geq j$. Thus the final string $z_1 z_2 \ldots$ is 1 2 5 6 9 11 13 14 17 19 ..., containing all odd numbers $< 2n$ except 3, 7, 15, 31, ...

(b) Similarly, the preorder permutation that characterizes the final tree is $2^k\, 2^{k-1} \ldots 1\,3\,5\,6\,7\,9\,10 \ldots$, where $k = \lfloor \lg n \rfloor$. Forestwise, node 2^j is the parent of 2^{j-1} nodes $\{2^{j-1}, 2^{j-1}+1, \ldots, 2^j - 1\}$, for $1 < j \leq k$, and the trees $\{2^k+1, \ldots, n\}$ are trivial.

Note: If Algorithm N is restarted at step N2 after it has terminated, it will generate the same sequence, but backwards. Algorithm L has the same property.

24. $l_0 l_1 \ldots l_{15} = 2\,0\,1\,0\,3\,0\,0\,6\,5\,0\,8\,0\,0\,12\,11\,4$; $r_1 \ldots r_{15} = 0\,15\,0\,10\,7\,0\,0\,9\,0\,14\,13\,0\,0\,0\,0$; $k_1 \ldots k_{15} = 0\,0\,2\,2\,4\,5\,5\,4\,8\,4\,10\,11\,11\,10\,2$; $q_1 \ldots q_{15} = 2\,1\,15\,4\,3\,10\,8\,5\,7\,6\,9\,14\,11\,13\,12$; and $u_1 \ldots u_{15} = 1\,2\,3\,1\,0\,0\,5\,0\,3\,1\,0\,0\,1\,0\,1\,0$. (If nodes of the forest F are numbered in postorder, k_j is the left sibling of j; or, if j is the leftmost child of p, $k_j = k_p$. Stated another way, k_j is the parent of j in the forest F^{TR}. And k_j is also $j-1-u_{n+1-j}$, the number of elements to the left of j in $q_1 \ldots q_n$ that are less than j.)

25. Taking a cue from Algorithms N and R, we want to extend each $(n-1)$-node tree to a list of two or more n-node trees. The idea in this case is to make n a child of $n-1$ in the binary tree at the beginning and the end of every such list. The following algorithm uses additional link fields p_j and s_j, where p_j points to the parent of j in the forest, and s_j points to j's left sibling or to j's rightmost sibling if j is the leftmost in its family. (These pointers p_j and s_j are, of course, not the same as the permutations $p_1 \ldots p_n$ in Table 1 or the scope coordinates $s_1 \ldots s_n$ in Table 2. In fact $s_1 \ldots s_n$ is the permutation λ of exercise 33 below.)

M1. [Initialize.] Set $l_j \leftarrow j+1$, $r_j \leftarrow 0$, $s_j \leftarrow j$, $p_j \leftarrow j-1$, and $o_j \leftarrow -1$ for $1 \leq j \leq n$, except that $l_n \leftarrow 0$.

M2. [Visit.] Visit $l_1 \ldots l_n$ and $r_1 \ldots r_n$. Then set $j \leftarrow n$.

M3. [Find j.] If $o_j > 0$, set $k \leftarrow p_j$ and go to M5 if $k \neq j-1$. If $o_j < 0$, set $k \leftarrow s_j$ and go to M4 if $k \neq j-1$. If $k = j-1$ in either case, set $o_j \leftarrow -o_j$, $j \leftarrow j-1$, and repeat this step.

M4. [Transfer down.] (At this point k is j's left sibling, or the rightmost member of j's family.) If $k \geq j$, terminate if $j = 1$, otherwise set $x \leftarrow p_j$, $l_x \leftarrow 0$, $z \leftarrow k$, and $k \leftarrow 0$ (thereby detaching node j from its parent and heading for the top level). But if $k < j$, set $x \leftarrow p_j + 1$, $z \leftarrow s_x$, $r_k \leftarrow 0$, and $s_x \leftarrow k$ (thereby detaching node j from k and going down a level). Then set $x \leftarrow k+1$, $y \leftarrow s_x$, $s_x \leftarrow z$, $s_j \leftarrow y$, $r_y \leftarrow j$, and $x \leftarrow j$. While $x \neq 0$, set $p_x \leftarrow k$ and $x \leftarrow r_x$. Return to M2.

M5. [Transfer up.] (At this point k is j's parent.) Set $x \leftarrow k+1$, $y \leftarrow s_j$, $z \leftarrow s_x$, $s_x \leftarrow y$, and $r_y \leftarrow 0$. If $k \neq 0$, set $y \leftarrow p_k$, $r_k \leftarrow j$, $s_j \leftarrow k$, $s_{y+1} \leftarrow z$, and $x \leftarrow j$; otherwise set $y \leftarrow j-1$, $l_y \leftarrow j$, $s_j \leftarrow z$, and $x \leftarrow j$. While $x \neq 0$, set $p_x \leftarrow y$ and $x \leftarrow r_x$. Return to M2. ▮

Running time notes: We can argue as in exercise 44 that step M3 costs $2C_n+3(C_{n-1}+\cdots+C_1)$ mems, and that steps M4 and M5 together cost $8C_n-2(C_{n-1}+\cdots+C_1)$, plus twice the number of times $x \leftarrow r_x$. The latter quantity is difficult to analyze precisely; for example, when $n=15$ and $j=6$, the algorithm sets $x \leftarrow r_x$ exactly $(1,2,3,4,5,6)$ times in respectively $(45,23,7,9,2,4)$ cases. But heuristically the average number of times $x \leftarrow r_x$ should be approximately $2-2^{j-n}$ when j is given, therefore about $(2C_n-(C_n-C_{n-1})-(C_{n-1}-C_{n-2})/2-(C_{n-2}-C_{n-3})/4-\cdots)/C_n \approx 8/7$ overall. Empirical tests confirm this predicted behavior, showing that the total cost per tree approaches $265/21 \approx 12.6$ mems as $n \to \infty$.

26. (a) The condition is clearly necessary. And if it holds, we can uniquely construct F: Node 1 and its siblings are the roots of the forest, and their descendants are defined inductively by noncrossing partitions. (In fact, we can compute the depth coordinates $c_1 \ldots c_n$ directly from Π's restricted growth sequence $a_1 \ldots a_n$: Set $c_1 \leftarrow 0$ and $i_0 \leftarrow 0$. For $2 \le j \le n$, if $a_j > \max(a_1,\ldots,a_{j-1})$, set $c_j \leftarrow c_{j-1}+1$ and $i_{a_j} \leftarrow c_j$, otherwise set $c_j \leftarrow i_{a_j}$.)

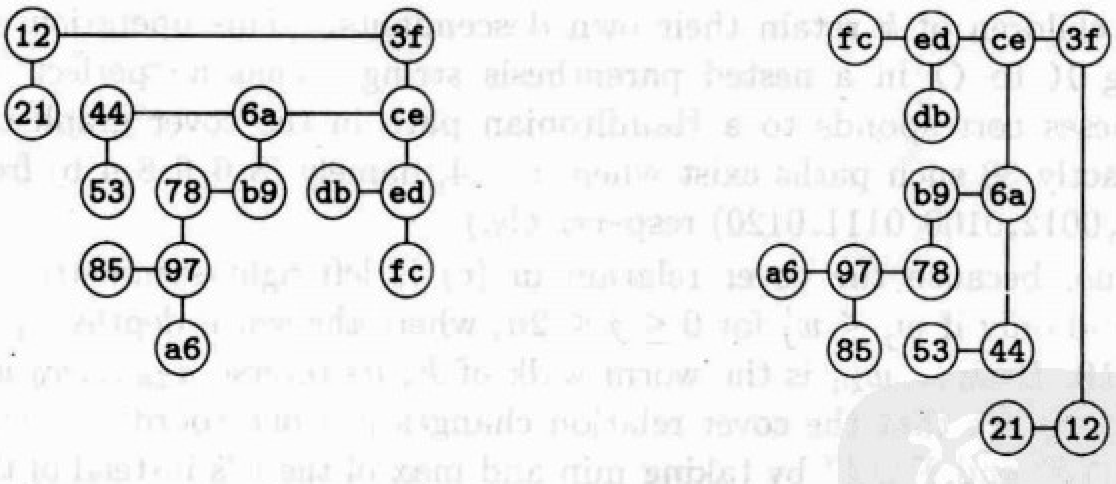
(b) If Π and Π' satisfy the noncrossing condition, so does their greatest common refinement $\Pi \vee \Pi'$, so we can proceed as in exercise 7.2.1.5–12(a).

(c) Let $x_1, \ldots, x_m$ be the children of some node in F, and let $1 \le j < k \le m$. Form F' by removing $x_{j+1}, \ldots, x_k$ from their family and reattaching them as children of $x_{j+1}-1$, the rightmost descendant of x_j.

(d) Obvious, by (c). Thus the forests are ranked from bottom to top by the number of nonleaf nodes they contain (which is one less than the number of blocks in Π).

(e) Exactly $\sum_{k=0}^{n} e_k(e_k-1)/2$, where $e_0 = n-e_1-\cdots-e_n$ is the number of roots.

(f) Dualization is similar to the transposition operation in exercise 12, but we use left-sibling and right-child links instead of left-child and right-sibling, and we transpose about the *minor* diagonal:



("Right" links now point downward. Notice that j is the rightmost child of k in F if and only if j is the left sibling of k in F^D. Preorder of F^D reverses the preorder of F, just as postorder of F^T reverses postorder of F.)

(g) From (f) we can see that F' covers F if and only if F^D covers F'^D. (Therefore F^D has $n+1-k$ leaves if F has k.)

(h) $F \barwedge F' = (F^D \veebar F'^D)^D$.

(i) No. If it did, equality would necessarily hold, by duality. But, for example, $0101 \barwedge 0121 = 0000$ and $0101 \veebar 0121 = 0123$, while leaves(0101) + leaves(0121) $\ne$ leaves(0000) + leaves(0123).

[Noncrossing partitions were first considered by H. W. Becker in *Math. Mag.* **22** (1948), 23–26. G. Kreweras proved in 1971 that they form a lattice; see the references in answer 2.3.4.6–3.]

27. (a) This assertion is equivalent to exercise 2.3.3–19.

(b) If we represent a forest by right-child and left-sibling links, preorder corresponds to inorder of the binary tree (see exercise 2.3.2–5), and s_j is the size of node j's right subtree. Rotation to the left at any nonleaf of this binary tree decreases exactly one of the scope coordinates, and the amount of decrease is as small as possible consistent with a valid table $s_1 \ldots s_n$. Therefore F' covers F if and only if F is obtained from F' by such a rotation. (Rotation in the left-child/right-sibling representation is similar, but with respect to postorder.)

(c) Dualization preserves the covering relation but exchanges left with right.

(d) $F \top F' = (F^D \perp F'^D)^D$. Equivalently, as noted in exercise 6.2.3–32, we can independently minimize the left-subtree sizes.

(e) The covering transformation in answer 26(c) obviously makes $s_j \le s'_j$ for all j.

(f) True, because $F \barwedge F' \ltimes F \dashv F \perp F'$ and $F \barwedge F' \ltimes F' \dashv F \perp F'$.

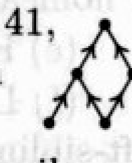
(g) False; for example, $0121 \veebar 0122 = 0123$ and $0121 \top 0122 = 0122$. (But we do have $F \top F' \dashv F \veebar F'$, by taking duals in (f).)

(h) The longest path, of length $\binom{n}{2}$, repeatedly decreases the rightmost nonzero s_j by 1. The shortest, of length $n-1$, repeatedly sets the leftmost nonzero s_j to 0.

Answer 6.2.3–32 gives many references to the literature of Tamari lattices.

28. (a) Just compute $\min(c_1, c'_1) \ldots \min(c_n, c'_n)$ and $\max(c_1, c'_1) \ldots \max(c_n, c'_n)$, because $c_1 \ldots c_n$ is a valid sequence if and only if $c_1 = 0$ and $c_j \le c_{j-1} + 1$ for $1 < j \le n$.

(b) Obvious because of (a). *Note:* The elements of any distributive lattice can be represented as the order ideals of some partial ordering. In the case of Fig. 41, that partial ordering is shown at the right, and a similar triangular grid with sides of length $n-2$ yields Stanley's lattice of order n.



(c) Take a node k of F that has a left sibling, j. Remove k from its family and place it as a new right child of j, followed by its former children as new children of j; the former children of k retain their own descendants. (This operation corresponds to changing)(to () in a nested parenthesis string. Thus a "perfect" Gray code for parentheses corresponds to a Hamiltonian path in the cover graph of Stanley's lattice. Exactly 38 such paths exist when $n = 4$, namely $(8, 6, 6, 8, 4, 6)$ from 0123 to $(1001, 0010, 0012, 0100, 0111, 0120)$ respectively.)

(d) True, because the cover relation in (c) is left-right symmetric. (We have $F \subseteq F'$ if and only if $w_j \le w'_j$ for $0 \le j \le 2n$, where the worm depths w_j are defined in exercise 10. If $w_0 \ldots w_{2n}$ is the worm walk of F, its reverse $w_{2n} \ldots w_0$ is the worm walk of F^R. Notice that the cover relation changes just one coordinate w_j. One can compute $F \cap F'$ and $F \cup F'$ by taking min and max of the w's instead of the c's.)

(e) See exercise 9. (Thus $F \perp F' \subseteq F \cap F'$, etc., as in exercise 27(f).)

Notes: Stanley introduced this lattice in *Fibonacci Quarterly* **13** (1975), 222–223. Since three important lattices are defined on the same elements, we need three notations for the different orderings; the symbols $\ltimes$, $\dashv$, and $\subseteq$ adopted here are intended to be reminiscent of the names of Kreweras, Tamari, and Stanley (who is Стенли in Russia).

29. If we paste six regular pentagons together, we get 14 vertices whose coordinates after suitable rotation and scaling are respectively

$$p_{1010} = p_{\bar{0}000} = p_{\overset{*}{3}000} = p_{\overset{*}{2}\bar{1}00} = (-1, \sqrt{3}, 2/\phi);$$
$$p_{0010} = p_{\overset{*}{3}100} = (\phi^{-2}, \sqrt{3}\,\phi, 0); \quad p_{3010} = p_{\bar{0}100} = (0, 0, 2); \quad p_{3210} = p_{\bar{0}200} = (2, 0, 2/\phi);$$
$$p_{0210} = p_{\overset{*}{3}200} = (\sqrt{5}, \sqrt{3}, 0); \quad p_{1000} = p_{\overset{*}{2}000} = (-\phi^2, \sqrt{3}/\phi, 0);$$

here $(x,y,z)^*$ means $(x,-y,z)$ and $(x,y,z)^-$ means $(x,y,-z)$. But then the three 4-edged "faces" are not squares; in fact, they don't even lie in a plane.

(One can however get a similar-looking solid, with true squares but irregular pentagons, by gluing together two suitable tetrahedra and lopping off the three glued-together corners. Alternative sets of coordinates for the associahedron, of substantial mathematical interest but less appealing to the eye, are discussed by Günter Ziegler in his *Lectures on Polytopes* (New York: Springer, 1995), example 9.11.)

30. (a) $\bar{f}_{n-1}\ldots\bar{f}_1\,0$, because internal node j in symmetric order has a nonempty right subtree if and only if internal node $j+1$ in symmetric order has an empty left subtree.

(b) In general if the footprint were $1^{p_1}0^{q_1+1}1^{p_2+1}0^{q_2+1}\ldots1^{p_k+1}0^{q_k+1}$, we would want to count all binary trees whose nodes in symmetric order have the specification $R^{p_1}NL^{q_1}BR^{p_2}NL^{q_2}B\ldots R^{p_k}NL^{q_k}$, where B means "both subtrees are nonempty," R means "the right subtree is nonempty but not the left," L means "the left subtree is nonempty but not the right," and N means "neither subtree is nonempty." This number in general is

$$\binom{p_1+q_1}{p_1}\binom{p_2+q_2}{p_2}\ldots\binom{p_k+q_k}{p_k}C_{k-1},$$

and in particular it is $\binom{1+0}{1}\binom{0+0}{0}\binom{1+0}{1}\binom{5+3}{5}\binom{0+0}{0}\binom{0+0}{0}\binom{0+2}{0}\binom{0+0}{0}\binom{1+2}{1}C_8=240240$.

(c) $d_j=0$ if and only if $c_{j+1}>c_j$, by exercise 3.

(d) In general, the footprint of $F\perp F'$ is $f_1\ldots f_n\wedge f'_1\ldots f'_n$, by exercise 27(a); the footprint of $F\top F'$ is $f_1\ldots f_n\vee f'_1\ldots f'_n$, by (a) and exercise 27(d).

[The fact that complements always exist in the Tamari lattice is due to H. Lakser; see G. Grätzer, *General Lattice Theory* (1978), exercise I.6.30.]

31. (a) 2^{n-1}; see exercise 6.2.2–5.

(b) $c_1\le\cdots\le c_n$; $d_1,\ldots,d_{n-1}\le1$; $e_j>0$ implies $e_j+\cdots+e_n=n-j$; $k_{j+1}\le k_j+1$; $p_1\le\cdots\le p_j\ge\cdots\ge p_n$ for some j; $s_j>0$ implies $s_j=n-j$; $u_1\ge\cdots\ge u_n$; $z_{j+1}\le z_j+2$. (Other constraints, which apply in general, whittle down the number of possibilities to 2^{n-1} in each case. For example, $u_1\ldots u_n$ must be a valid sequence of scope coordinates.)

(c) True in only n cases out of 2^{n-1}. (But F^T *is* degenerate.)

(d) The degenerate forest with footprint $f_1\ldots f_n$ has $c_{j+1}=c_j+f_j$. Elements $j<k$ are siblings if and only if $f_j=f_{j+1}=\cdots=f_{k-1}=0$. Thus if F'' is the degenerate forest with footprint $f_1\ldots f_n\wedge f'_1\ldots f'_n$, then $F''\ltimes F$ and $F''\ltimes F'$; hence $F''\ltimes F\barwedge F'\dashv F\perp F'$. And we also have $F\perp F'\dashv F''$ by (b). A similar argument proves that $F\veebar F'=F\top F'$ is the degenerate forest with footprint $f_1\ldots f_n\vee f'_1\ldots f'_n$.

Thus, when the Kreweras and Tamari lattices are restricted to degenerate forests, they become identical to the Boolean lattice of subsets of $\{1,\ldots,n-1\}$. [This result, in the case of Tamari lattices, is due to George Markowsky, *Order* **9** (1992), 265–290, whose paper also shows that Tamari lattices enjoy many further properties.]

32. Suppose F and F' have scope coordinates $s_1\ldots s_n$ and $s'_1\ldots s'_n$. Call index j *frozen* if $s_j<s'_j$ or $j=0$. We want to specify the values of the frozen coordinates and maximize the others. Let $s_0=n$, and for $0\le k\le n$ let

$$s''_k=s_j-k+j,\quad\text{where } j=\max\{i\mid 0\le i\le k,\ i\text{ is frozen, and } i+s_i\ge k\}.$$

Since $s_k\le s_j-(k-j)$ whenever $0\le k-j\le s_j$, we have $s''_k\ge s_k$, with equality when k is frozen.

The scopes $s''_0 s''_1 \ldots s''_n$ correspond to a valid forest according to the condition of exercise 27(a). For if $k \geq 0$ and $0 \leq l \leq s''_k = s_j - k + j$ and $s''_{k+l} = s_{j'} - k - l + j'$, we have $s''_{k+l} + l \leq s''_k$ if $0 \leq j' - j \leq s_j$, because $s_{j'} + j' - j \leq s_j$ in that case. And we can't have $j > j'$ or $j' > j + s_j$, because $j + s_j \geq k + l \geq j'$.

Let F''' be a forest with scopes satisfying $s_k \leq s'''_k \leq s''_k$. Then $\min(s'_k, s'''_k) = s_k$, because $s_k = s''_k$ when k is frozen, otherwise $s_k = s'_k$.

Conversely, if F''' is a forest with $F' \perp F''' = F$, we must have $s_k \leq s'''_k \leq s''_k$. For $s'''_k < s_k$ would imply $s'''_k < s'_k$. And if k is minimal with $s'''_k > s''_k$, we have $s''_k = s_j - k + j$ for some frozen j with $0 \leq j \leq k$ and $j + s_j \geq k$. Then $s'''_j \geq s_j$ implies $k - j \leq s'''_j$, hence $s'''_k + k - j \leq s'''_j$. If $j < k$ we have $s'''_j \leq s''_j = s_j$, a contradiction. But $j = k$ implies $\min(s'''_k, s'_k) > s_k$.

To get the first semidistributive law, apply this principle with F replaced by $F \perp G$ and F' replaced by F; then the hypotheses $F \dashv G \dashv F''$ and $F \dashv H \dashv F''$ imply that $F \dashv G \top H \dashv F''$. The second semidistributive law follows by taking duals in the first.

(Ralph Freese suggests calling F'' the *pseudo-complement* of F' over F.)

33. (a) Let $k\lambda =$ `LLINK[k]` if `LLINK[k]` $\neq 0$, otherwise `RLINK[k-1]` if $k \neq 1$, otherwise the root of the binary tree. This rule defines a permutation because $k\lambda = j$ if and only if $k = \text{parent}(j) + [j \text{ is a right child}]$, or $k = 1$ and j is the root. Also $k\lambda \geq k$ when `LLINK[k]` $= 0$ and $k\sigma\lambda \leq k$ when `RLINK[k]` $= 0$. [For a generalization to t-ary trees, see P. H. Edelman, *Discrete Math.* **40** (1982), 171–179.]

(b) Using the representation of (2) in answer 26(f), we see that $\lambda(F)$ is $(3\,1)(2)(12\,6\,4)(5)(11\,7)(14\,13)(9\,8)(15)(10)$ in that case. In general the cycles are the families of the forest, in decreasing order within each cycle; nodes are numbered in preorder. [See Dershowitz and Zaks, *Discrete Math.* **62** (1986), 215–218.]

(c) $\lambda(F^D) = \rho\sigma\lambda\rho$, where ρ is the "flip" permutation $(1\,n)(2\,n{-}1)\ldots$, because the dual forest interchanges `LLINK` $\leftrightarrow$ `RLINK` and flips the preorder numbering.

(d) The cycle breakup $(x_j\,x_k)(x_1 \ldots x_m) = (x_1 \ldots x_j x_{k+1} \ldots x_m)(x_{j+1} \ldots x_k)$ corresponds to answer 26(c).

(e) By (d), each covering path corresponds to a factorization of $(n \ldots 2\,1)$. Let q_n denote the number of such factorizations. Then we have the recurrence $q_1 = 1$ and $q_n = \sum_{l=1}^{n-1}(n-l)\binom{n-2}{l-1}q_l q_{n-l}$, because there are $n-l$ choices with $k - j = l$ by which the first transposition breaks the cycle into parts of sizes l and $n-l$, then $\binom{n-2}{l-1}$ ways to interleave the subsequent factors. The solution is $q_n = n^{n-2}$, because

$$\sum_{l=1}^{n-1}\binom{n-1}{l} l^{l-1}(y-l)^{n-1-l} = \lim_{x\to 0}\sum_{l=1}^{n-1}\binom{n-1}{l}(x+l)^{l-1}(y-l)^{n-1-l}$$
$$= \lim_{x\to 0}\frac{(x+y)^{n-1} - y^{n-1}}{x} = (n-1)y^{n-1}.$$

[See J. Dénes, *Magyar Tudományos Akadémia Matematikai Kutató Intézetének Közleményei* **4** (1959), 63–70. It is natural to seek a correspondence between factorizations and labeled free trees, since there also happen to be n^{n-2} of the latter. Perhaps the simplest is the following, given $(1\,2 \ldots n) = (x_1\,y_1)\ldots(x_{n-1}\,y_{n-1})$ where $x_j < y_j$: Suppose the cycle containing x_j and y_j in $(x_j\,y_j)\ldots(x_{n-1}\,y_{n-1})$ is $(z_1 \ldots z_m)$, where $z_1 < \cdots < z_m$. If $y_j = z_m$, let $a_j = z_1$, otherwise let $a_j = \min\{z_i \mid z_i > x_j\}$. Then one can show that $a_1 \ldots a_{n-1}$ is a "wake-up sequence" for parking $n-1$ cars, and exercise 6.4–31 connects it to free trees.]

34. Each covering path from bottom to top is equivalent to a Young tableau of shape $(n-1, n-2, \ldots, 1)$, so we can use Theorem 5.1.4H. (See exercise 5.3.4–38.)

[The enumeration of such paths in Tamari lattices remains mysterious; the relevant sequence is 1, 1, 2, 9, 98, 2981, 340549, ...]

35. Multiply by $n+1$, then see *AMM* **97** (1990), 626–630.

36. We might as well generalize to t-ary trees for arbitrary $t \ge 1$, by making obvious amendments to steps T1–T5. Let $C_n^{(t)}$ be the number of t-ary trees with n internal nodes; thus $C_n = C_n^{(2)}$ and $C_n^{(t)} = ((t-1)n+1)^{-1}\binom{tn}{n}$. If h of the degrees b_j are changed between visits, we have $h \ge x$ in $C_{n-x}^{(t)}$ cases. So the easy case occurs with probability $1 - C_{n-1}^{(t)}/C_n^{(t)} \approx 1 - (t-1)^{t-1}/t^t$, and the average number of times $b_j \leftarrow 0$ in step T4 is $(C_{n-1}^{(t)} + \cdots + C_1^{(t)})/C_n^{(t)} \approx (t-1)^{t-1}/(t^t - (t-1)^{t-1})$, or 4/23 when $t = 3$.

Indeed, we can also study the t-ary recursive structure $A_{pq}^{(t)} = 0\,A_{p(q-1)}^{(t)},\ t\,A_{(p-1)q}^{(t)}$ when $0 \le (t-1)p \le q \ne 0$, generalizing (5). The number of such degree sequences, $C_{pq}^{(t)}$, satisfies the recurrence (21) except that $C_{pq}^{(t)} = 0$ when $p < 0$ or $(t-1)p > q$. The general solution is

$$C_{pq}^{(t)} = \frac{q-(t-1)p+1}{q+1}\binom{p+q}{p} = \binom{p+q}{p} - (t-1)\binom{p+q}{p-1},$$

and we have $C_n^{(t)} = C_{n((t-1)n)}^{(t)}$. The triangle for $t = 3$ begins as shown at the right.

1				
1				
1	1			
1	2			
1	3	3		
1	4	7		
1	5	12	12	
1	6	18	30	
1	7	25	55	55
1	8	33	88	143

37. The basic lexicographic recursion for all such forests is

$$A(n_0, n_1, \ldots, n_t) = 0\,A(n_0-1, n_1, \ldots, n_t),$$
$$1\,A(n_0, n_1-1, \ldots, n_t),\ \ldots,\ t\,A(n_0, n_1, \ldots, n_t-1)$$

when $n_0 > n_2 + 2n_3 + \cdots + (t-1)n_t$ and $n_1, \ldots, n_t \ge 0$; otherwise $A(n_0, n_1, \ldots, n_t)$ is empty, except that $A(0, \ldots, 0) = \epsilon$ is the sequence consisting of the empty string alone. Step G1 computes the first entry of $A(n_0, \ldots, n_t)$. We want to analyze five quantities:

C, the number of times G2 is executed (the total number of forests);
E, the number of times G3 goes to G2 (the number of easy cases);
K, the number of times G4 moves some b_i into list c;
L, the number of times G5 compares b_j with some c_i;
Z, the number of times G5 sets $c_1 \leftarrow 0$.

Then the loop in step G6 sets $b_{l-c_j} \leftarrow c_j$ a total of $K - Z - n_1 - \cdots - n_t$ times.

Let n be the vector $(n_0, n_1, \ldots, n_t)$, and let e_j be the unit vector with 1 in coordinate position j. Let $|n| = n_0 + n_1 + \cdots + n_t$ and $\|n\| = n_1 + 2n_2 + \cdots + tn_t$. Using this notation we can rewrite the basic recurrence above in the convenient form

$$A(n) = 0\,A(n-e_0),\ 1\,A(n-e_1),\ \ldots,\ t\,A(n-e_t) \qquad \text{when } |n| > \|n\|.$$

Consider the general recurrence relation

$$F(n) = f(n) + \Bigl(\sum_{j=0}^{t} F(n-e_j)\Bigr)\bigl[|n| > \|n\|\bigr],$$

with $F(n) = 0$ whenever the vector n has a negative component. If $f(n) = [|n|=0]$, then $F(n) = C(n)$ is the total number of forests. Answer 2.3.4.4–32 tells us that

$$C(n) = \frac{(|n|-1)!\,(|n|-\|n\|)}{n_0!\,n_1!\ldots n_t!} = \sum_{j=0}^{t}(1-j)\binom{|n|-1}{n_0,\ldots,n_{j-1},n_j-1,n_{j+1},\ldots,n_t}.$$

generalizing the formula for $C_{pq}^{(t)}$ in answer 26 (which is the case $n_0=(t-1)q+1$ and $n_t=p$). Similarly, we obtain recurrences for the other quantities $E(n)$, $K(n)$, $L(n)$, and $Z(n)$ needed in our analysis by choosing other kernel functions $f(n)$:

$$\begin{aligned}
f(n) &= [|n| = n_0+1 \text{ and } n_0 > \|n\|] &&\text{yields}\quad F(n)=E(n);\\
f(n) &= [|n| > n_0] &&\text{yields}\quad F(n)=E(n)+K(n);\\
f(n) &= [|n| = \|n\|+1] &&\text{yields}\quad F(n)=C(n)+K(n)-Z(n);\\
f(n) &= \textstyle\sum_{1\le j<k\le t} n_j[n_k>0] &&\text{yields}\quad F(n)=L(n).
\end{aligned}$$

The symbolic methods of exercise 2.3.4.4–32 do not seem to yield quick solutions to these more general recurrences, but we can readily establish the value of $C-E$ by noting that $b_m+m<N$ in step G2 if and only if the previous step was G3. Therefore

$$C(n)-E(n) = \sum_{j=1}^{t} C(n-f_j), \qquad \text{where } f_j = e_j-(j-1)e_0;$$

this sum counts the subforests in which $n_1+\cdots+n_t$, the number of internal (nonleaf) nodes, has decreased by 1. Similarly we can let

$$C^{(x)}(n) = \sum\{C(n-i_1f_1-\cdots-i_tf_t) \mid i_1+\cdots+i_t = x\}$$

be the number of subforests having $n_1+\cdots+n_t-x$ internal nodes. Then we have

$$K(n)-Z(n) = \sum_{x=1}^{|n|} C^{(x)}(n),$$

a formula analogous to (20), because $k-[b_j=0]\ge x\ge 1$ in step G5 if and only if $b_{m-x}>0$ and $b_{m-x+1}\ge\cdots\ge b_m$. Such preorder degree strings are in one-to-one correspondence with the forests of $C^{(x)}(n)$ if we remove $b_{m-x+1}\ldots b_m$ and an appropriate number of trailing 0s from the string $b_1\ldots b_N$.

From these formulas we can conclude that the Zaks–Richards algorithm needs only $O(1)$ operations per forest visited, whenever $n_1 = n_2+\cdots+n_t+O(1)$, because $C(n-f_j)/C(n) = n_j n_0^{\underline{j-1}}/(|n|-1)^{\underline{j}} \le 1/4+O(|n|^{-1})$ when $j>1$. Indeed, the value of K is quite small in nearly all cases of practical interest. However, the algorithm can be slow when n_1 is large. For example, if $t=1$, $n_0=m+r+1$, and $n_1=m$, the algorithm essentially computes all r-combinations of $m+r$ things; then $C(n)=\binom{m+r}{r}$ and $K(n)-Z(n)=\binom{m+r}{r+1}=\Omega(mC(n))$ when r is fixed. [To ensure efficiency i all cases, we can keep track of trailing 1s; see Ruskey and Roelants van Baronaigien, *Congressus Numerantium* **41** (1984), 53–62.]

Exact formulas for K, Z, and (especially) L do not seem to be simple, but we can compute those quantities as follows. Say that the "active block" of a forest is the rightmost substring of nonzero degrees; for example, the active block of 302102021230000000 is 2123. All permutations of the active block occur equally often. Indeed, let $D(n)$ denote the sum of "trailing zeros$(\beta)-1$" over all preorder degree strings β for forests

of specification n. Then a block with n'_j occurrences of j for $1 \le j \le t$ is active in exactly $D(n - n'_1 f_1 - \cdots - n'_t f_t) + [n'_1 + \cdots + n'_t = n_1 + \cdots + n_t]$ cases. For example, given the string 3021020000, we can insert 21230000 in three places to obtain a forest with active block 2123. The contributions to K and L when the active block is flush left (not preceded by any 0s) can be computed as in exercise 7.2.1.2–6, namely

$$k(n) = w(e_{n_1}(z) \ldots e_{n_t}(z)), \qquad l(n) = w\Big(e_{n_1}(z) \ldots e_{n_t}(z) \sum_{1 \le i < j \le t} (n_i - z r_i(z)) r_j(z)\Big)$$

in the notation of that answer. Analogous contributions occur in general; therefore

$$K(n) = k(n) + \sum D(n-n')k(n'),\ L(n) = l(n) + \sum D(n-n')l(n'),\ Z(n) = \sum D(n-n'),$$

summed over all vectors n' such that $n'_j \le n_j$ for $1 \le j \le t$ and $|n'| - \|n'\| = |n| - \|n\|$ and $n'_1 + \cdots + n'_t \le n_1 + \cdots + n_t - 2$.

It remains to determine $D(n)$. Let $C(n;j)$ be the number of forests of specification $n = (n_0, \ldots, n_t)$ in which the last internal node in preorder has degree j. Then we have

$$C(n) = \sum_{j=1}^{t} C(n;j) \text{ and } C(n+e_1;1) = C(n+e_2;2) = \cdots = C(n+e_t;t) = C(n) + D(n).$$

From this infinite system of linear equations we can deduce that $C(n) + D(n)$ is

$$\sum_{i_2=0}^{n_2} \cdots \sum_{i_t=0}^{n_t} (-1)^{i_2+\cdots+i_t} \binom{i_2 + \cdots + i_t}{i_2, \ldots, i_t} C(n + (1+i_2+\cdots+i_t)e_1 - i_2 f_2 - \cdots - i_t f_t).$$

Simpler expressions would of course be desirable, if they exist.

38. Step L1 obviously uses $4n+2$ mems. Step L3 exits to L4 or L5 exactly $C_j - C_{j-1}$ times with a particular value of j; therefore it costs $2C_n + 3\sum_{j=0}^{n}(n-j)(C_j - C_{j-1}) = 2C_n + 3(C_{n-1} + \cdots + C_1 + C_0)$ mems. Steps L4 and L5 jointly cost a total of $6C_n - 6$. Therefore the entire process involves $9 + O(n^{-1/2})$ mems per visit.

39. A Young tableau of shape (q,p) and entries y_{ij} corresponds to an element of A_{pq} that has left parens in positions $p+q+1-y_{21}, \ldots, p+q+1-y_{2p}$ and right parens in positions $p+q+1-y_{11}, \ldots, p+q+1-y_{1q}$. The hook lengths are $\{q+1, q, \ldots, 1, p, p-1, \ldots, 1\} \setminus \{q-p+1\}$; so $C_{pq} = (p+q)!(q-p+1)/(p!(q+1)!)$ by Theorem 5.1.4H.

40. (a) $C_{pq} = \binom{p+q}{p} - \binom{p+q}{p-1} \equiv \binom{p+q}{p} + \binom{p+q}{p-1} = \binom{p+q+1}{p}$ (modulo 2); now use exercise 1.2.6–11. (b) By Eq. 7.1–(oo) we know that $\nu(n \,\&\, (n+1)) = \nu(n+1) - 1$.

41. It equals $C(wz)/(1 - zC(wz)) = 1/(1 - z - wzC(wz)) = (1 - wC(wz))/(1 - w - z)$, where $C(z)$ is the Catalan generating function (18). The first of these formulas, $C(wz) + zC(wz)^2 + z^2C(wz)^3 + \cdots$, is easily seen to be equivalent to (24). [See P. A. MacMahon, *Combinatory Analysis* **1** (Cambridge Univ. Press, 1915), 128–130.]

42. (a) Elements $a_1 \ldots a_n$ determine an entire self-conjugate nested string $a_1 \ldots a_{2n}$, and there are $C_{q(n-q)}$ possibilities for $a_1 \ldots a_n$ having exactly q right parentheses. So the answer is

$$\sum_{q=0}^{\lfloor n/2 \rfloor} C_{q(n-q)} = \sum_{q=0}^{\lfloor n/2 \rfloor} \left(\binom{n}{q} - \binom{n}{q-1}\right) = \binom{n}{\lfloor n/2 \rfloor}.$$

(b) Exactly $C_{(n-1)/2}\,[n \text{ odd}]$, because a self-transpose binary tree is determined by its left subtree. And (c) has the same answer, because F is self-dual if and only if F^R is self-transpose.

43. $C_{pq} = C_q - \binom{q-p-1}{1}C_{q-1} + \cdots = \sum_{r=0}^{q-p}(-1)^r\binom{q-p-r}{r}C_{q-r}$, by induction on $q-p$.

44. The number of mems between visits is $3j-2$ in step B3, $h+1$ in step B4, and 4 in step B5, where h is the number of times $y \leftarrow r_y$. The number of binary trees with $h \geq x$, given j and x, is $[z^{n-j-x-1}]\,C(z)^{x+3}$ when $j < n$, because we get such trees by attaching $x+3$ subtrees below $j+x+1$ internal nodes. Setting $x=0$ tells us that a given value of j occurs $C_{(n-j-1)(n-j+1)} = C_{n+1-j} - C_{n-j}$ times, using (24) and exercise 43. Thus $\sum j$ over all binary trees is $n + \sum_{j=1}^{n}(C_{n+1-j} - C_{n-j})j = C_n + C_{n-1} + \cdots + C_1$. Similarly, $\sum(h+1)$ is $\sum_{j=1}^{n-1}\sum_{x=0}^{n-j-1} C_{(n-j-x-1)(n-j+1)} = \sum_{j=1}^{n-1} C_{(n-j-1)(n-j+2)} = \sum_{j=1}^{n}(C_{n-j+2} - 2C_{n-j+1}) = C_{n+1} - (C_n + C_{n-1} + \cdots + C_0)$. So overall, the algorithm costs $C_{n+1} + 4C_n + 2(C_{n-1} + \cdots + C_1) + O(n) = (26/3 - 10/(3n) + O(n^{-2}))C_n$ mems.

45. Each of the easy cases in step K3 occurs C_{n-1} times, so the total cost of that step is $3C_{n-1} + 8C_{n-1} + 2(C_n - 2C_{n-1})$ mems. Step K4 fetches r_i a total of $[z^{n-i-1}]\,C(z)^{i+2} = C_{(n-i-1)n}$ times; summing for $i \geq 2$ gives $C_{(n-3)(n+1)} = C_{n+1} - 3C_n + C_{n-1}$ mems altogether in that loop. Step K5 costs $6C_n - 12C_{n-1}$. Step K6 is a bit more complicat d, but one can show that the operation $r_j \leftarrow r_k$ is performed $C_n - 3C_{n-1} + 1$ times when $n > 2$, while the operation $r_j \leftarrow 0$ is performed $C_{n-1} - n + 1$ times. The total number of mems therefore comes to $C_{n+1} + 7C_n - 9C_{n-1} + n + 3 = (8.75 - 9.375/n + O(n^{-2}))C_n$.

Although this total is asymptotically worse than that of Algorithm B in answer 44, the large negative coefficient of n^{-1} means that Algorithm B actually wins only when $n \geq 58$; and n won't ever be that big.

46. (a) Going to the left from (pq) increases the area by $q-p$.

(b) The leftward steps on a path from (nn) to (00) correspond to the left parentheses in $a_1 \ldots a_{2n}$, and we have $q - p = c_k$ at the kth such step.

(c) Equivalently, $C_{n+1}(x) = \sum_{k=0}^{n} x^k C_k(x) C_{n-k}(x)$. This recurrence holds because an $(n+1)$-node forest F consists of the root of the leftmost tree together with a k-node forest F_l (the descendants of that root) and an $(n-k)$-node forest F_r (the remaining trees), and because we have

$$\text{internal path length}(F) = k + \text{internal path length}(F_l) + \text{internal path length}(F_r).$$

(d) The strings of $A_{p(p+r)}$ have the form $\alpha_0)\alpha_1)\ldots\alpha_{r-1})\alpha_r$ where each α_j is properly nested. The area of such a string is the sum over j of the area of α_j plus $r-j$ times the number of left parens in α_j.

Notes: The polynomials $C_{pq}(x)$ were introduced by L. Carlitz and J. Riordan in *Duke Math. J.* **31** (1964), 371–388; the identity in part (d) is equivalent to their formula (10.12). They also proved that

$$C_{pq}(x) = \sum_r (-1)^r x^{r(r-1)-\binom{q-p}{2}} \binom{q-p-r}{r}_x C_{q-r}(x),$$

generalizing the result of exercise 43. From part (c) we have the infinite continued fraction $C(x,z) = 1/(1 - z/(1 - xz/(1 - x^2z/(1 - \cdots))))$, which G. N. Watson proved is equal to $F(x,z)/F(x,z/x)$, where

$$F(x,z) = \sum_{n=0}^{\infty} \frac{(-1)^n x^{n^2} z^n}{(1-x)(1-x^2)\ldots(1-x^n)};$$

see *J. London Math. Soc.* **4** (1929), 39–48. We have already encountered the same generating function, slightly disguised, in exercise 5.2.1–15.

The internal path length of a forest is the "left path length" of the corresponding binary tree, namely the sum over all internal nodes of the number of left branches on the path from the root. The more general polynomial

$$C_n(x,y) = \sum x^{\text{left path length}(T)} y^{\text{right path length}(T)},$$

summed over all n-node binary trees T, seems to obey no simple additive recurrence like the one for $C_{nn}(x) = C_n(x,1)$ studied in this exercise; but we do have $C_{n+1}(x,y) = \sum_k x^k C_k(x,y) y^{n-k} C_{n-k}(x,y)$. Therefore the super generating function $C(x,y,z) = \sum_n C_n(x,y)z^n$ satisfies the functional equation $C(x,y,z) = 1 + zC(x,y,xz)C(x,y,yz)$. (The case $x=y$ was considered in exercise 2.3.4.5–5.)

47. $C_n(x) = \sum_q x^{\binom{q-p}{2}} C_{pq}(x) C_{(n-q)(n-1-p)}(x)$ for $0 \le p < n$.

48. Let $\bar{C}(z) = C(-1,z)$ in the notation of exercise 46, and let $\bar{C}(z)\bar{C}(-z) = F(z^2)$. Then $\bar{C}(z) = 1 + zF(z^2)$ and $\bar{C}(-z) = 1 - zF(z^2)$; so $F(z) = 1 - zF(z)^2$, and $F(z) = C(-z)$. It follows that $C_{pq}(-1) = [z^p]\, C(-z^2)^{\lceil (q-p)/2\rceil}(1 + zC(-z^2))^{[q-p\text{ even}]}$, which is $(-1)^{(p/2)} C_{(p/2)(q/2-1)}[p\text{ even}]$ when q is even, $(-1)^{\lfloor p/2\rfloor} C_{\lfloor p/2\rfloor\lfloor q/2\rfloor}$ when q is odd. A perfect Gray code through the strings of A_{pq} can exist only if $|C_{pq}(-1)| \le 1$, because the associated graph is bipartite (see Fig. 41); $|C_{pq}(-1)|$ is the difference between the sizes of the parts, because each perfect transposition changes $c_1 + \cdots + c_n$ by ± 1.

49. By Algorithm U with $n = 15$ and $N = 10^6$, it is `()(()())((()()))((((()()))))`.

50. Make the following changes to Algorithm U: In step U1, also set $r \leftarrow 0$. In step U3, test if $a_m =$ ')' instead of testing if $N \le c'$. In step U4, set $r \leftarrow r + c'$ instead of $N \leftarrow N - c'$. And omit the assignments to a_m in steps U3 and U4.

The string in (1) turns out to have rank 3141592. (Who knew?)

51. By Theorem 7.2.1.3L, $N = \binom{\bar{z}_1}{n} + \binom{\bar{z}_2}{n-1} + \cdots + \binom{\bar{z}_n}{1}$; hence $\kappa_n N = \binom{\bar{z}_1}{n-1} + \binom{\bar{z}_2}{n-2} + \cdots + \binom{\bar{z}_n}{0}$, since $\bar{z}_n \ge 1$. Now note that $N - \kappa_n N$ is the rank of $z_1 z_2 \ldots z_n$, because of (23) and exercise 50. (For example, let $z_1 \ldots z_4 = 1256$, which has rank 6 in Table 1. Then $\bar{z}_1 \ldots \bar{z}_4 = 7632$, $N = 60$, and $\kappa_4 60 = 54$. Notice that N is fairly large, because $\bar{z}_1 = 2n-1$; Fig. 27 shows that $\kappa_n N$ usually *exceeds* N when N is smaller.)

52. The number of trailing right parentheses has the same distribution as the number of leading left parentheses, and the sequence of nested strings that begin with '$(^k)$' is $(^k)\,A_{(n-k)(n-1)}$. Therefore the probability that $d_n = k$ is $C_{(n-k)(n-1)}/C_n$. We find

$$\sum_{k=0}^{n} \binom{k}{t} C_{(n-k)(n-1)} = \sum_{k=0}^{n} \left(\binom{2n-1-k}{n-1} - \binom{2n-1-k}{n}\right)\binom{k}{t}$$
$$= \binom{2n}{n+t} - \binom{2n}{n+t+1} = C_{(n-t)(n+t)}$$

using Eq. 1.2.6–(25), and it follows that the mean and variance are respectively equal to $3n/(n+2) = 3 - 6/(n+2)$ and $2n(2n^2-n-1)/((n+2)^2(n+3)) = 4 + O(n^{-1})$. [The moments of this distribution were first calculated by R. Kemp in *Acta Informatica* **35** (1998), 17–89, Theorem 9. Notice that $c_n = d_n - 1$ has essentially the same behavior.]

53. (a) $3n/(n+2)$, by exercise 52. (b) H_n, by exercise 6.2.2–7. (c) $2 - 2^{-n}$, by induction.

(d) Any particular (but fixed) sequence of left or right branches has the same distribution of steps before a leaf is encountered. (In other words, the probability that a node with Dewey binary notation 01101 occurs is the same as the probability that 00000 occurs.) Thus if $X = k$ with probability p_k, each of the 2^k potential nodes on

level k is external with probability p_k. The expected value $\sum_k 2^k p_k$ is therefore the expected number of external nodes, namely $n+1$ in all three cases. (One can of course also verify this result directly, with $p_k = C_{(n-k)(n-1)}/C_n$ in case (a), $p_k = \left[{n \atop k}\right]/n!$ in case (b), and $p_k = 2^{-k+[k=n]}$ in case (c).)

Notes: The average path length turns out to be $\Theta(\sqrt{n})$, $\Theta(\log n)$, and $\Theta(n)$ in these three cases; thus it is longer when the expected time to hit a leaf is shorter! The reason is that ubiquitous "holes" near the root force other paths to be long. Case (a) has an interesting generalization to t-ary trees, when $p_k = C^{(t)}_{(n-k)((t-1)n-1)}/C^{(t)}_n$ in the notation of answer 36. Then the mean distance to a leaf is $(t+1)n/((t-1)n+2)$, and it is instructive to prove via telescoping series that

$$\sum_k t^k C^{(t)}_{(n-k)((t-1)n-1)} = \binom{tn}{n}.$$

54. Differentiating with respect to x we have

$$C'(x,z) = zC'(x,z)C(x,xz) + zC(x,z)(C'(x,xz) + zC_l(x,xz)),$$

where $C_l(x,z)$ denotes the derivative of $C(x,z)$ with respect to z. Thus $C'(1,z) = 2zC'(1,z)C(z) + z^2C(z)C'(z)$; and since $C'(z) = C(z)^2 + 2zC(z)C'(z)$ we can solve for $C'(1,z)$, obtaining $z^2C(z)^3/(1-2zC(z))^2$. Therefore $\sum(c_1+\cdots+c_n) = [z^n]\,C'(1,z) = 2^{2n-1} - \frac{1}{2}(3n+1)C_n$, in agreement with exercise 2.3.4.5–5. Similarly we find

$$\sum(c_1+\cdots+c_n)^2 = [z^n]\,C''(1,z) = \Bigl(\frac{5n^2+19n+6}{6}\Bigr)\binom{2n}{n} - \Bigl(1+\frac{3n}{2}\Bigr)4^n.$$

Thus the mean and variance are $\frac{1}{2}\sqrt{\pi}n^{3/2} + O(n)$ and $(\frac{5}{6}-\frac{\pi}{4})n^{3/2} + O(n)$, respectively.

55. Differentiating as in answer 54, and using the formulas of exercises 46(d) and 5.2.1–14 together with $[z^n]\,C(z)^r/(1-4z) = 2^{2n+r} - \sum_{j=1}^r 2^{r-j}\binom{2n+j}{n}$, yields

$$\begin{aligned}C'_{p(p+r)}(1) &= [z^p]\Bigl((r+1)\frac{z^2C(z)^{r+3}}{1-4z} + \binom{r+1}{2}\frac{zC(z)^{r+2}}{\sqrt{1-4z}}\Bigr)\\ &= [z^p]\Bigl((r+1)\frac{C(z)^{r+1}-2C(z)^r+C(z)^{r-1}}{1-4z} + \binom{r+1}{2}\frac{C(z)^{r+1}-C(z)^r}{\sqrt{1-4z}}\Bigr)\\ &= (r+1)\Bigl(2^{2p+r-1} - \binom{2p+r+1}{p} - \sum_{j=1}^{r-1}2^{r-1-j}\binom{2p+j}{p}\Bigr) + \binom{r+1}{2}\binom{2p+r}{p-1}.\end{aligned}$$

56. Use 1.2.6–53(b). [See *BIT* **30** (1990), 67–68.]

57. $2S_0(a,b) = \binom{2a}{a}\binom{2b}{b} + \binom{2a+2b}{a+b}$ by 1.2.6–(21). Exercise 1.2.6–53 tells us that

$$\sum_{k=a-m}^{a}\binom{2a}{a-k}\binom{2b}{b-k}k = (m+1)(a+b-m)\binom{2a}{m+1}\binom{2b}{a+b-m};$$

therefore $2S_1(a,b) = \binom{2a}{a}\binom{2b}{b}\frac{ab}{a+b}$. And since $b^2S_p(a,b) - S_{p+2}(a,b) = S_p(a,b-1)$, we find $2S_2(a,b) = \binom{2a+2b}{a+b}\frac{ab}{2a+2b-1}$; $2S_3(a,b) = \binom{2a}{a}\binom{2b}{b}a^2b^2/(a+b)^2$. Formula (30) follows by setting $a=m$, $b=n-m$, and $C_{(x-k)(x+k)} = \binom{2x}{x-k} - \binom{2x}{x-k-1}$.

Similarly, the average of w_{2m-1} is $\sum_{k\ge 0}(2k-1)C_{(m-k)(m+k-1)}C_{(n-m-k+1)(n-m+k)}$ divided by C_n, namely

$$\frac{2S_3(m,n+1-m) - S_2(m,n+1-m)}{m(n+1-m)C_n} = \frac{m(n+1-m)}{n}\binom{2m}{m}\binom{2n+2-2m}{n+1-m}\Big/\binom{2n}{n} - 1.$$

[R. Kemp, *BIT* **20** (1980), 157–163; H. Prodinger, *Soochow J. Math.* **9** (1983), 193–196.]

58. Summing over cases in which the left subtree has k internal nodes, we have

$$t_{lmn} = [l=m=n=0] + \sum_{k=0}^{m-1} C_k t_{(l-1)(m-k-1)(n-k-1)} + \sum_{k=m}^{n-1} C_{n-1-k} t_{(l-1)mk}.$$

Thus the triple generating function $t(v,w,z) = \sum_{l,m,n} t_{lmn} v^l w^m z^n$ satisfies

$$t(v,w,z) = 1 + vwzC(wz)t(v,w,z) + vzC(z)t(v,w,z);$$

and the analogous linear relation for $t(w,z) = \partial t(v,w,z)/\partial v\,|_{v=1}$ follows, because $t(1,w,z) = \sum_{n=0}^{\infty}\sum_{m=0}^{n} C_n w^m z^n = (C(z) - wC(wz))/(1-w)$ and $zC(z)^2 = C(z) - 1$. Algebraic manipulation now yields

$$t(w,z) = \frac{C(z) + wC(wz) - (1+w)}{(1-w)^2 z} - \frac{2wC(z)C(wz)}{(1-w)^2} - \frac{C(z) - wC(wz)}{1-w},$$

and we obtain the formula $t_{mn} = (m+1)C_{n+1} - 2\sum_{k=0}^{m}(m-k)C_k C_{n-k} - C_n$. Now

$$\sum_{k=0}^{m-1}(k+1)C_k C_{n-1-k} = \frac{m}{2n}\binom{2m}{m}\binom{2n-2m}{n-m}$$

can be proved as in exercise 56, and it follows that

$$t_{mn} = 2\binom{2m}{m}\binom{2n-2m}{n-m}\frac{(2m+1)(2n-2m+1)}{(n+1)(n+2)} - C_n, \qquad \text{for } 0 \le m \le n.$$

[P. Kirschenhofer, *J. Combinatorics, Information and System Sciences* **8** (1983), 44–60. For higher moments and generalizations, see W. J. Gutjahr, *Random Structures and Algorithms* **3** (1992), 361–374; A. Panholzer and H. Prodinger, *J. Statistical Planning and Inference* **101** (2002), 267–279. Note that the generating function $t(v,w,z)$ yields

$$t_{lmn} = \sum_k \binom{l}{k} C_{(m-k)(m-1)} C_{(n-m-l+k)(n-m-1)}.$$

Using the fact that $\sum_k \binom{k}{r} C_{(n-k)(m-1)} = C_{(n-r)(m+r)}$ when $m \ge 1$, we obtain the formula $t_{mn} + C_n = \sum_k (k+1) C_{(m-k)(m-1)} C_{(n-m)(n-m+k+1)}$, a sum that can therefore (surprisingly) be expressed in closed form.]

59. $$T(w,z) = \frac{w(C(z)-C(wz))}{(1-w)} - wzC(z)C(wz) + zC(z)T(w,z) + wzC(wz)T(w,z)$$
$$= \frac{w((C(z)+C(wz)-2)/z - (1+w)C(z)C(wz) - (1-w)(C(z)-C(wz)))}{(1-w)^2}.$$

Hence $T_{mn} = t_{mn} - \sum_{k=m}^{n} C_k C_{n-k}$. [Is there a combinatorial proof?] And

$$T_{mn} = \binom{2m}{m}\binom{2n+2-2m}{n+1-m}\frac{4m(n+1-m) + n + 1}{2(n+1)(n+2)} - \frac{1}{2}C_{n+1} - C_n, \quad \text{for } 1 \le m \le n.$$

60. (a) It is the number of right parentheses in co-atoms. (Therefore it is also the number of k for which $w_{2k-1} < 0$ in the associated "worm walk.")

(b) For convenience let d('(') $= +1$ and d(')') $= -1$.

A1. [Initialize.] Set $i \leftarrow j \leftarrow 1$ and $k \leftarrow 2n$.

A2. [Done?] Terminate the algorithm if $j > k$. Otherwise set $a_j \leftarrow$ '(', $j \leftarrow j+1$.

A3. [Atom?] If $b_i =$ ')', set $s \leftarrow -1$, $i \leftarrow i+1$, and go to A4. Otherwise set $s \leftarrow 1$, $i \leftarrow i+1$, and while $s > 0$ set $a_j \leftarrow b_i$, $j \leftarrow j+1$, $s \leftarrow s+d(b_i)$, $i \leftarrow i+1$. Return to A2.

A4. [Co-atom.] Set $s \leftarrow s+d(b_i)$. Then if $s < 0$, set $a_k \leftarrow b_i$, $k \leftarrow k-1$, $i \leftarrow i+1$, and repeat step A4. Otherwise set $a_k \leftarrow$ ')', $k \leftarrow k-1$, $i \leftarrow i+1$, and return to A2. ▌

(c) The defect-11 inverse of (1) is `(()))((())))))(()((())(()))(((`. In general we find it by locating the subscript m just before the lth-from-last right parenthesis, and the indices $(u_0, v_0), \ldots, (u_{s-1}, v_{s-1})$ of matching parentheses such that $u_j \le m < v_j$.

I1. [Initialize.] Set $c \leftarrow j \leftarrow s \leftarrow 0$, $k \leftarrow m \leftarrow 2n$, and $u_0 \leftarrow 2n+1$.

I2. [Scan right to left.] If $a_k =$ ')', go to I3; if $a_k =$ '(', go to I4; if $k = 0$, go to I5.

I3. [Process a ')'.] Set $r_j \leftarrow k$, $j \leftarrow j+1$, $c \leftarrow c+1$. If $c = l$, set $m \leftarrow k-1$, $s \leftarrow j$, and $u_s \leftarrow k$. Then decrease k by 1 and return to I2.

I4. [Process a '('.] (At this point the left parenthesis a_k matches the right parenthesis $a_{r_{j-1}}$.) Set $j \leftarrow j-1$. If $r_j > m$, set $u_j \leftarrow k$ and $v_j \leftarrow r_j$. Then decrease k by 1 and return to I2.

I5. [Prepare to permute.] Set $i \leftarrow j \leftarrow 1$, $k \leftarrow 2n$, and $c \leftarrow 0$.

I6. [Permute.] While $j \ne u_c$, set $b_i \leftarrow a_j$, $i \leftarrow i+1$, $j \leftarrow j+1$. Then terminate if $c = s$; otherwise set $b_i \leftarrow$ ')', $i \leftarrow i+1$, $j \leftarrow j+1$. While $k \ne v_c$, set $b_i \leftarrow a_k$, $i \leftarrow i+1$, $k \leftarrow k-1$. Then set $b_i \leftarrow$ '(', $i \leftarrow i+1$, $k \leftarrow k-1$, $c \leftarrow c+1$, and repeat step I6. ▌

Notes: The fact that exactly C_n balanced strings of length $2n$ have defect l, for $0 \le l \le n$, was discovered by P. A. MacMahon [*Philosophical Transactions* **209** (1909), 153–175, §20], then rediscovered by K. L. Chung and W. Feller [*Proc. Nat. Acad. Sci.* **35** (1949), 605–608], using generating functions. A simple combinatorial explanation was found subsequently by J. L. Hodges, Jr. [*Biometrika* **42** (1955), 261–262], who observed that if $\beta_1 \ldots \beta_r$ has defect $l > 0$ and if $\beta_k = \alpha_k^R$ is its rightmost co-atom, the balanced string $\beta_1 \ldots \beta_{k-1}(\beta_{k+1} \ldots \beta_r)\alpha_k'^R$ has defect $l-1$ (and this transformation is reversible). The efficient mapping in the present exercise is similar to a construction of M. D. Atkinson and J.-R. Sack [*Information Processing Letters* **41** (1992), 21–23].

61. (a) Let $c_j = 1 - b_j$; thus $c_j \le 1$, $c_1 + \cdots + c_N = f$, and we must prove that

$$c_1 + c_2 + \cdots + c_k < f \quad \text{if and only if} \quad k < N$$

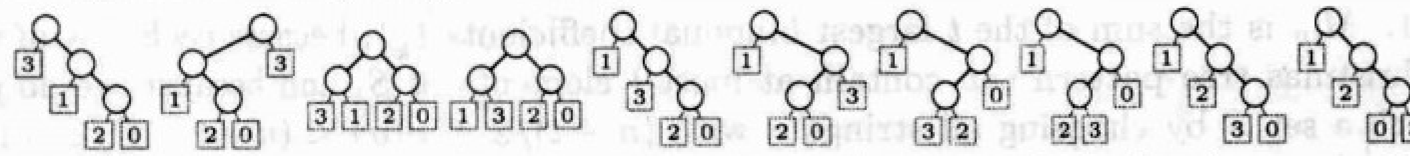
holds for exactly f cyclic shifts. We can define c_j for all integers j by letting $c_{j \pm N} = c_j$. Let us also define Σ_j for all j by letting $\Sigma_0 = 0$ and $\Sigma_j = \Sigma_{j-1} + c_j$; then $\Sigma_{j+Nt} = \Sigma_j + ft$, and $\Sigma_{j+1} \le \Sigma_j + 1$. It follows that for each integer x there is a smallest integer $j = j(x)$ such that $\Sigma_j = x$. Moreover, $j(x) < j(x+1)$; and $j(x+f) = j(x) + N$. Thus the desired condition holds if and only if we shift by $j(x) \bmod N$ for $x = 1, 2, \ldots,$ or f. (The history of this important lemma is discussed in answer 2.3.4.4–32.)

(b) Start with $l \leftarrow m \leftarrow s \leftarrow 0$. Then for $k = 1, 2, \ldots, N$ (in this order) do the following: Set $s \leftarrow s + 1 - b_k$; and if $s > m$, set $m \leftarrow s$, $j_l \leftarrow k$, and $l \leftarrow (l+1) \bmod f$. The answers are $j_0, \ldots, j_{f-1}$, by the proof in part (a).

(c) Start with any string $b_1 b_2 \ldots b_N$ containing n_j occurrences of j for $0 \le j \le t$. Apply a random permutation to this string, then apply the algorithm of part (b). Choose randomly between $(j_0, \ldots, j_{f-1})$ and use the resulting cyclic shift as a preorder sequence to define the forest.

[See L. Alonso, J. L. Rémy, and R. Schott, *Algorithmica* **17** (1997), 162–182. for an even more general algorithm.]

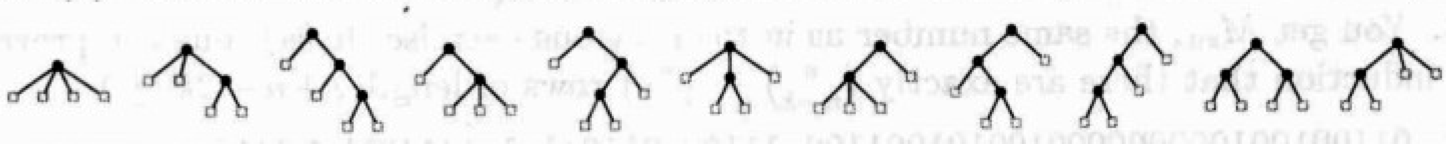
62. Bit strings $(l_1 \ldots l_n, r_1 \ldots r_n)$ are valid if and only if $b_1 \ldots b_n$ is valid in exercise 20, where $b_j = l_j + r_j$. Therefore we can use exercise 61. [See J. F. Korsh, *Information Processing Letters* **45** (1993), 291–294.]

63. 

64. $X = 2k + b$ where $(k, b) = (0,1)$, $(2,1)$, $(0,0)$, $(5,1)$, $(6,0)$, $(1,1)$; eventually $L_0L_1 \ldots L_{12} = 5\ 11\ 3\ 4\ 0\ 7\ 9\ 8\ 1\ 6\ 10\ 12\ 2$.

65. See A. Panholzer and H. Prodinger, *Discrete Mathematics* **250** (2002), 181–195; M. Lučak and P. Winkler, *Random Structures and Algorithms* **24** (2004), 420–443.

66. (a) "Shrink" the white edges, merging the nodes that they connect. For example,



are the ordinary trees that correspond to the eleven Schröder trees depicted for $n = 3$. Under this correspondence a left link means, "here is a child"; a white right link means, "look here for more children"; a black right link means, "here's the last child."

(b) Mimic Algorithm L, but between rotations use an ordinary Gray binary code to run through all color patterns of whatever right links are present. (The case $n = 3$ has, in fact, been illustrated in the example.)

Note that Schröder trees also correspond to series-parallel graphs, as in (53). They do, however, impose an order on the edges and/or superedges that are joined in parallel; so they correspond more precisely to series-parallel graphs *as embedded in the plane* (and with edges and vertices unlabeled, except for s and t).

67. $S(z) = 1 + zS(z)(1 + 2(S(z) - 1))$, because $1 + 2(S(z) - 1)$ enumerates the right subtrees; therefore $S(z) = (1 + z - \sqrt{1 - 6z + z^2}\,)/(4z)$.

Notes: We've seen Schröder numbers in exercise 2.3.4.4–31, where $G(z) = zS(z)$; and in exercise 2.2.1–11, where $b_n = 2S_{n-1}$ for $n \ge 2$ and where we found the recurrence $(n-1)S_n = (6n-3)S_{n-1} - (n-2)S_{n-2}$. They grow asymptotically as explored in exercise 2.2.1–12. A triangle of numbers S_{pq}, analogous to (22), can be used to generate random Schröder trees. These numbers satisfy

$$\begin{aligned} S_{pq} &= S_{p(q-1)} + S_{(p-1)q} + S_{(p-2)q} + \cdots + S_{0q} = S_{p(q-1)} + 2S_{(p-1)q} - S_{(p-1)(q-1)} \\ &= \frac{q-p+1}{q+1} \sum_{k=0}^{p} \binom{q+1}{p-k}\binom{p-1}{k} 2^k = \sum_{k=0}^{p} \left(\binom{q}{p-k}\binom{p-1}{k} - \binom{q}{p-k-1}\binom{p-1}{k-1} \right) 2^k \\ &= [w^p z^q]\, S(wz)/(1 - zS(wz)); \end{aligned}$$

the double generating function on the last line is due to Emeric Deutsch. Many other properties of Schröder trees are discussed in Richard Stanley's *Enumerative Combinatorics* **2** (1999), exercise 6.39.

68. A single row that contains only the empty string ϵ. (The general rule (36) for going from $n-1$ to n converts this row into '0 1', the pattern of order 1.)

69. The first $\binom{6}{3} = 20$ rows are the Christmas tree pattern of order 6, if we ignore the '10' at the beginning of each string. The pattern of order 7 is a bit more difficult to see; but there are $\binom{7}{3} = 35$ rows in which the leftmost entry begins with 0. Disregard the rightmost string in all such rows, and ignore the 0 at the beginning of each remaining string. (Other answers are also possible.)

70. If σ appears in column k of the Christmas tree pattern, let σ' be the string in column $n-k$ of the same row. (If we think of parentheses instead of bits, this rule takes the mirror reflection of the free parentheses in the sense of answer 11, by (39).)

71. M_{tn} is the sum of the t largest binomial coefficients $\binom{n}{k}$, because each row of the Christmas tree pattern can contain at most t elements of S, and because we do get such a set S by choosing all strings σ with $(n-t)/2 \le \nu(\sigma) \le (n+t-1)/2$. (The formula

$$M_{tn} = \sum_{n-t\le 2k\le n+t-1} \binom{n}{k}$$

is about as simple as possible; however, special formulas like $M_{(2)n} = M_{n+1}$ hold for small t, and we also have $M_{tn} = 2^n$ for $t > n$.)

72. You get M_{sn}, the same number as in the previous exercise. In fact, one can prove by induction that there are exactly $\binom{n}{n-k} - \binom{n}{k-s}$ rows of length $s+n-2k \ge 0$.

73. 011001001000000000100101001100, 111001011011111111101101011100; see (38).

74. By the lexicographic property, we want to count the number of rows whose rightmost elements have the respective forms $0*^{29}$, $10*^{28}$, $110*^{27}$, $111000*^{24}$, $1110010*^{22}$, $111001010*^{21}$, $1110010110 0*^{19}$, $11100101101 0*^{18}$, $111001011011 0*^{17}$, ..., namely all 30-bit strings that precede $\tau = 111001011011111111101101011100$.

If θ has p more 1s than 0s, the number of Christmas tree rows ending with $\theta *^n$ is the same as the number of rows ending with $1^p *^n$; and this is $M_{(p+1)n}$, by exercise 71, because all such rows are the n-step descendants of the starting row '$0^p\ 0^{p-1}1\ \ldots\ 1^p$'.

Consequently the answer is $M_{0(29)} + M_{1(28)} + M_{2(27)} + M_{1(24)} + \cdots + M_{(12)3} + M_{(13)2} = \sum_{k=1}^{21} M_{(2k-1-z_k)(n-z_k)} = 0 + \binom{28}{14} + \binom{27}{14} + \binom{27}{13} + \binom{24}{12} + \cdots + 8 + 4 = 84867708$, where $(z_1, \ldots, z_{21}) = (1, 2, 3, 6, \ldots, 27, 28)$ is the sequence of places where 1s occur in τ.

75. We have $r_1^{(n)} = M_{n-2}$, because row $r_1^{(n)}$ is the bottom descendant of the first row in (33). We also have $r_{j+1}^{(n)} - r_j^{(n)} = M_{j(n-1-j)} - M_{(j-1)(n-2-j)} = M_{(j+1)(n-2-j)}$ by the formula in answer 74, because the relevant sequence $z_1 \ldots z_{n-1}$ for row $r_j^{(n)}$ is $1^j 0 1^{n-1-j}$. Therefore, since $M_{jn}/M_n \to j$ for fixed j as $n \to \infty$, we have

$$\lim_{n\to\infty} \frac{r_j^{(n)}}{M_n} = \sum_{k=1}^{j} \frac{k}{2^{k+1}} = 1 - \frac{j+2}{2^{j+1}}.$$

And we've also implicitly proved that $\sum_{k=0}^{n} M_{k(n-k)} = M_{n+1} - 1$.

76. The first $\binom{2n}{n}$ elements of the infinite sequence

$$Q = 1313351313351335355713133513133513353557131335133535571335355735575779\ldots$$

are the row sizes in the pattern of order $2n$; this sequence $Q = q_1q_2q_3\ldots$ is the unique fixed point of the transformation that maps $1 \mapsto 13$ and $n \mapsto (n-2)\,n\,n\,(n+2)$ for odd $n > 1$, representing two steps of (36).

Let $f(x) = \limsup_{n\to\infty} s(\lceil xM_n \rceil)/n$ for $0 < x \le 1$. This function apparently vanishes almost everywhere; but it equals 1 when x has the form $(q_1 + \cdots + q_j)/2^n$, because of answer 72. On the other hand if we define $g(x) = \lim_{n\to\infty} s(\lceil xM_n \rceil)/\sqrt{n}$, the function $g(x)$ appears to be measurable, with $\int_0^1 g(x)\,dx = \sqrt{\pi}$, although $g(x)$ is infinite when $f(x) > 0$. (Rigorous proofs or disproofs of these conjectures are solicited.)

77. The hint follows from (39), by considering worm walks; so we can proceed thus:

X1. [Initialize.] Set $a_j \leftarrow 0$ for $0 \le j \le n$; also set $x \leftarrow 1$. (In the following steps we will have $x = 1 + 2(a_1 + \cdots + a_n)$.)

X2. [Correct the tail.] While $x \le n$, set $a_x \leftarrow 1$ and $x \leftarrow x + 2$.

X3. [Visit.] Visit the bit string $a_1 \ldots a_n$.

X4. [Easy case?] If $a_n = 0$, set $a_n \leftarrow 1$, $x \leftarrow x + 2$, and return to X3.

X5. [Find and advance a_j.] Set $a_n \leftarrow 0$ and $j \leftarrow n-1$. Then while $a_j = 1$, set $a_j \leftarrow 0$, $x \leftarrow x-2$, and $j \leftarrow j-1$. Stop if $j = 0$; otherwise set $a_j \leftarrow 1$ and go back to X2. ▮

78. True, by (39) and exercise 11.

79. (a) List the indices of the 0s, then the indices of the 1s; for instance, the bit string in exercise 73 corresponds to the permutation 1 4 5 7 8 10 11 12 13 20 23 25 29 30 2 3 6 9 14 15 16 17 18 19 21 22 24 26 27 28.

(b) Using the conventions of (39), the P tableau has the indices of left parentheses and free parentheses in its top row, other indices in the second row. Thus, from (38),

$P =$

1	2	3	6	8	9	11	12	13	14	15	16	17	18	19	21	22	24	26	27	28
4	5	7	10	20	23	25	29	30												

.

[See K.-P. Vo, *SIAM J. Algebraic and Discrete Methods* **2** (1981), 324–332, for a generalization to chains of submultisets.]

80. This curious fact is a consequence of exercise 79 together with Theorem 6 in the author's paper on tableaux; see *Pacific J. Math.* **34** (1970), 709–727.

81. Suppose σ and σ' belong respectively to chains of length s and s' in the Christmas tree patterns of order n and n'. At most $\min(s, s')$ of the ss' pairs of strings in those chains can be in the biclutter. Furthermore, because of (39), those ss' pairs of strings actually constitute exactly $\min(s, s')$ chains in the Christmas tree pattern of order $n + n'$, when they are concatenated. Therefore the sum of $\min(s, s')$ over all pairs of chains is $M_{n+n'}$, and the result follows. We have incidentally proved the nonobvious identity

$$\sum_{j,k} \min(m+1-2j, n+1-2k)\, C_{j(m-j)} C_{k(n-k)} = M_{m+n}.$$

Notes: This extension of Sperner's theorem was proved independently by G. Katona [*Studia Sci. Math. Hungar.* **1** (1966), 59–63] and D. J. Kleitman [*Math. Zeitschrift* **90** (1965), 251–259]. See Greene and Kleitman, *J. Combinatorial Theory* **A20** (1976), 80–88, for the proof given here and for further results.

82. (a) There is at least one evaluation in each row m; there are two if and only if $s(m) > 1$ and the first evaluation yields 0. Thus if f is identically 1, we get the minimum, M_n; if f is identically 0, we get the maximum, $M_n + \sum_m [s(m) > 1] = M_{n+1}$.

(b) Let $f(\chi(m, n/2)) = 0$ in the $C_{n/2}$ cases where $s(m) = 1$; otherwise let $f(\chi(m, a)) = 1$, where a is defined by the algorithm. When n is odd, this rule implies that $f(\sigma)$ is always 1; but when n is even, $f(\sigma) = 0$ if and only if σ is first in its row. (To see why, use the fact that the row containing σ'_j in (41) always has size $s - 2$.) This function f is indeed monotonic; for if $\sigma \le \tau$ and if σ has a free left parenthesis, so does τ. For example, in the case $n = 8$ we have

$$f(x_1, \ldots, x_8) = x_8 \vee x_6x_7 \vee x_4x_5(x_6 \vee x_7) \vee x_2x_3(x_4(x_5 \vee x_6 \vee x_7) \vee x_5(x_6 \vee x_7)).$$

(c) In these circumstances (45) is the solution for all n.

83. At most 3 outcomes are possible in step H4 — in fact, at most 2 when $s(m) = 1$. [See exercise 5.3.4–31 for sharper bounds; in the notation of that exercise, there are exactly $\delta_n + 2$ monotone Boolean functions of n Boolean variables.]

84. For this problem we partition the 2^n bit strings into M_n blocks instead of chains, where the strings $\{\sigma_1, \ldots, \sigma_s\}$ of each block satisfy $\|A\sigma_i^T - A\sigma_j^T\| \geq 1$ for $i \neq j$; then at most one bit string per block can satisfy $\|A\sigma^T - b\| < \frac{1}{2}$.

Let A' denote the first $n-1$ columns of A, and let v be the nth column. Suppose $\{\sigma_1, \ldots, \sigma_s\}$ is a block for A', and number the subscripts so that $v^T A' \sigma_1^T$ is the minimum of $v^T A' \sigma_j^T$. Then rule (36) defines appropriate blocks for A, because we have $\|A(\sigma_i 0)^T - A(\sigma_j 0)^T\| = \|A(\sigma_i 1)^T - A(\sigma_j 1)^T\| = \|A'\sigma_i^T - A'\sigma_j^T\|$ and

$$\|A(\sigma_j 1)^T - A(\sigma_1 0)^T\|^2 = \|A'\sigma_j^T + v - A'\sigma_1^T\|^2$$
$$= \|A'(\sigma_j - \sigma_1)^T\|^2 + \|v\|^2 + 2v^T A'(\sigma_j - \sigma_1)^T \geq \|v\|^2 \geq 1.$$

[And more is true; see *Advances in Math.* **5** (1970), 155–157. This result extends a theorem of J. E. Littlewood and A. C. Offord, *Mat. Sbornik* **54** (1943), 277–285, who considered the case $m = 2$.]

85. If V has dimension $n - m$, we can renumber the coordinates so that

$$\begin{matrix} (1, & 0, & \ldots, & 0, & x_{11}, & \ldots, & x_{1m}) \\ (0, & 1, & \ldots, & 0, & x_{21}, & \ldots, & x_{2m}) \\ \vdots & \vdots & \ddots & \vdots & \vdots & & \vdots \\ (0, & 0, & \ldots, & 1, & x_{(n-m)1}, & \ldots, & x_{(n-m)m}) \end{matrix}$$

is a basis, with none of the row vectors $v_j = (x_{j1}, \ldots, x_{jm})$ entirely zero. Let $v_{n-m+1} = (-1, 0, \ldots, 0)$, …, $v_n = (0, 0, \ldots, -1)$. Then the number of 0–1 vectors in V is the number of 0–1 solutions to $Ax = 0$, where A is the $m \times n$ matrix with columns $v_1, \ldots, v_n$. But this quantity is at most the number of solutions to $\|Ax\| < \frac{1}{2}\min(\|v_1\|, \ldots, \|v_n\|)$, which is at most M_n by exercise 84.

Conversely, the basis with $m = 1$ and $x_{j1} = (-1)^{j-1}$ yields M_n solutions. [This result has application to electronic voting; see Golle's Ph.D. thesis (Stanford, 2004).]

86. First reorder the 4-node subtrees so that their level codes are 0121 (plus a constant); then sort larger and larger subtrees until everything is canonical. The resulting level codes are 0 1 2 3 4 3 2 1 2 3 2 1 2 0 1, and the parent pointers are 0 1 2 3 4 3 2 1 8 9 8 1 12 0 14.

87. (a) The condition holds if and only if $c_1 < \cdots < c_k \geq c_{k+1} \geq \cdots \geq c_n$ for some k, so the total number of cases is $\sum_k \binom{n-1}{n-k} = 2^{n-1}$.

(b) Note that $c_1 \ldots c_k = c'_1 \ldots c'_k$ if and only if $p_1 \ldots p_k = p'_1 \ldots p'_k$; and in such cases, $c_{k+1} < c'_{k+1}$ if and only if $p_{k+1} < p'_{k+1}$.

88. Exactly A_{n+1} forests are visited, and A_k of them have $p_k = \cdots = p_n = 0$. Therefore O4 is performed A_n times; and p_k is changed $A_{k+1} - 1$ times in step O5, for $1 \leq k < n$. Step O5 also changes p_n a total of $A_n - 1$ times. The average number of mems per visit is therefore only $2+3/(\alpha-1)+O(1/n) \approx 3.534$, if we keep p_n in a register. [See E. Kubicka, *Combinatorics, Probability and Computing* **5** (1996), 403–417.]

89. If step O5 sets $p_n \leftarrow p_j$ exactly Q_n times, it sets $p_k \leftarrow p_j$ exactly $Q_k + A_{k+1} - A_k$ times, for $1 < k < n$, because every prefix of a canonical $p_1 \ldots p_n$ is canonical. We have $(Q_1, Q_2, \ldots) = (0, 0, 1, 2, 5, 9, 22, 48, 118, 288, \ldots)$; and one can show that $Q_n = \sum_{d \geq 1} \sum_{1 \leq c < n/d-1} a_{(n-cd)(n-cd-d)}$, where a_{nk} is the number of canonical parent sequences $p_1 \ldots p_n$ with $p_n = k$. But these numbers a_{nk} remain mysterious.

90. (a) This property is equivalent to 2.3.4.4–(7); vertex 0 is the centroid.

(b) Let $m = \lfloor n/2 \rfloor$. At the end of step O1, set $p_{m+1} \leftarrow 0$, and also $p_{2m+1} \leftarrow 0$ if n is odd. At the end of step O4, set $i \leftarrow j$ and while $p_i \neq 0$ set $i \leftarrow p_i$. (Then i is the root of the tree containing j and k.) At the beginning of step O5, if $k = i + m$ and $i < j$, set $j \leftarrow i$ and $d \leftarrow m$.

(c) If n is even, there are no bicentroidal trees with $n+1$ vertices. Otherwise find all pairs $(p'_1 \ldots p'_m, p''_1 \ldots p''_m)$ of canonical forests on $m = \lfloor n/2 \rfloor$ nodes, with $p'_1 \ldots p'_m \geq p''_1 \ldots p''_m$; let $p_1 = 0$, $p_{j+1} = p'_j + 1$, and $p_{m+j+1} = (p''_j + m + 1)[p''_j > 0]$ for $1 \leq j \leq m$. (Two incarnations of Algorithm O will generate all such sequences. This algorithm for free trees is due to F. Ruskey and G. Li; see *SODA* **10** (1999), S939–S940.)

91. Use the following recursive procedure $W(n)$: If $n \leq 2$, return the unique n-node oriented tree. Otherwise choose positive integers j and d so that a given pair (j, d) is obtained with probability $dA_dA_{n-jd}/((n-1)A_n)$. Compute random oriented trees $T' \leftarrow W(n-jd)$ and $T'' \leftarrow W(d)$. Return the tree T obtained by linking j clones of T'' to the root of T'. [*Combinatorial Algorithms* (Academic Press, 1975), Chapter 25.]

92. Not always. [R. L. Cummins, in *IEEE Trans.* **CT-13** (1966), 82–90, proved that the graph of $S(G)$ always contains a cycle; see also C. A. Holzmann and F. Harary, *SIAM J. Applied Math.* **22** (1972), 187–193. But their constructions are unsuitable for efficient computation, because they require foreknowledge of the parity of the sizes of intermediate results.]

93. Yes. Step S7 undoes step S3; step S9 undoes the deletions of step S8.

94. For example, we can use depth-first search, with an auxiliary table $b_1 \ldots b_n$:

i) Set $b_1 \ldots b_n \leftarrow 0 \ldots 0$, then $v \leftarrow 1$, $w \leftarrow 1$, $b_1 \leftarrow 1$, and $k \leftarrow n-1$.

ii) Set $e \leftarrow n_{v-1}$. While $t_e \neq 0$, do the following substeps:
   - a) Set $u \leftarrow t_e$. If $b_u \neq 0$, go to substep (c).
   - b) Set $b_u \leftarrow w$, $w \leftarrow u$, $a_k \leftarrow e$, $k \leftarrow k-1$. Terminate if $k = 0$.
   - c) Set $e \leftarrow n_e$.

iii) If $w \neq 1$, set $v \leftarrow w$, $w \leftarrow b_w$, and return to (ii). Otherwise report an error: The given graph was not connected.

We could actually terminate as soon as substep (b) reduces k to 1, since Algorithm S never looks at the initial value of a_1. But we might as well test for connectivity.

95. The following steps perform a breadth-first search from u, to see if v is reachable without using edge e. An auxiliary array $b_1 \ldots b_n$ of arc pointers is used, which should be initialized to $0 \ldots 0$ at the end of step S1; we will reset it to $0 \ldots 0$ again.

i) Set $w \leftarrow u$ and $b_w \leftarrow v$.

ii) Set $f \leftarrow n_{u-1}$. While $t_f \neq 0$, do the following substeps:
   - a) Set $v' \leftarrow t_f$. If $b_{v'} \neq 0$, go to substep (d).
   - b) If $v' \neq v$, set $b_{v'} \leftarrow v$, $b_w \leftarrow v'$, $w \leftarrow v'$, and go to substep (d).
   - c) If $f \neq e \oplus 1$, go to step (v).
   - d) Set $f \leftarrow n_f$.

iii) Set $u \leftarrow b_u$. If $u \neq v$, return to step (ii).

iv) Set $u \leftarrow t_e$. While $u \neq v$, set $w \leftarrow b_u$, $b_u \leftarrow 0$, $u \leftarrow w$. Go to S9 (e is a bridge).

v) Set $u \leftarrow t_e$. While $u \neq v$, set $w \leftarrow b_u$, $b_u \leftarrow 0$, $u \leftarrow w$. Then set $u \leftarrow t_e$ again and continue step S8 (e is not a bridge).

Two quick heuristics can be used before starting this calculation: If $d_u = 1$, then e is obviously a bridge; and if $l_{l_e} \neq 0$, then e is obviously a nonbridge (because there's another edge between u and v). Such special cases are detected readily by the breadth-first

search, yet experiments by the author indicate that both heuristics are definitely worthwhile. For example, the test on l_{l_e} typically saves 3% or so of the total running time.

96. (a) Let e_k be the arc $k-1 \to k$. The steps in answer 94 set $a_k \leftarrow e_{n+1-k}$ for $n > k \geq 1$. Then at level k we shrink e_{n-k}, for $1 \leq k < n-1$. After visiting the (unique) spanning tree $e_{n-1} \ldots e_2 e_n$, we unshrink e_{n-k} and discover quickly that it is a bridge, for $n-1 > k \geq 1$. Thus the running time is linear in n; in the author's implementation it turns out to be exactly $102n - 226$ mems for $n \geq 3$.

However, this result depends critically on the order of the edges in the initial spanning tree. If step S1 had produced "organ-pipe order" such as

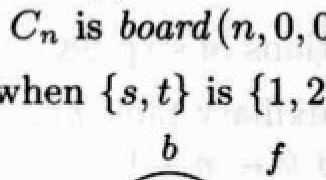
$$e_{n/2+1}\, e_{n/2}\, e_{n/2+2}\, e_{n/2-1} \ldots e_{n-1}\, e_2$$

in positions $a_2 \ldots a_{n-1}$ when n is even, the running time would have been $\Omega(n^2)$, because $\Omega(n)$ of the bridge tests would each have taken $\Omega(n)$ steps.

(b) Now a_k is initially e_{n-k} for $n > k \geq 1$, where e_1 is the arc $n \to 1$. The spanning trees visited, when $n \geq 4$, are respectively $e_{n-2} \ldots e_1 e_n$, $e_{n-2} \ldots e_1 e_{n-1}$, $e_{n-2} \ldots e_2 e_{n-1} e_n$, $e_{n-2} \ldots e_3 e_{n-1} e_n e_1$, $\ldots$, $e_{n-1} e_n e_1 \ldots e_{n-3}$. Following the tree $e_{n-2} \ldots e_{k+2} e_{n-1} e_n e_1 \ldots e_k$ the computations move down to level $n-k-3$ and up again, for $0 \leq k \leq n-4$; the bridge tests are all efficient. Thus the total running time is quadratic (in the author's version, exactly $35.5n^2 + 7.5n - 145$ mems, for $n \geq 5$).

Incidentally, P_n is *board*$(n,0,0,0,1,0,0)$ in the notation of the Stanford GraphBase, and C_n is *board*$(n,0,0,0,1,1,0)$; the SGB vertices are named 0 through $n-1$.

97. Yes, when $\{s,t\}$ is $\{1,2\}$, $\{1,3\}$, $\{2,3\}$, $\{2,4\}$, or $\{3,4\}$, but not $\{1,4\}$.

98. $A' =$ 

; this is the "dual planar graph" of the planar graph A.

(The near trees of A' are complements of the spanning trees of A, and vice versa.)

99. The stated method works, by induction on the size of the tree, for essentially the same reasons that it worked for n-tuples in Section 7.2.1.1 — but with the additional proviso that we must successively designate each child of an uneasy node.

Leaf nodes are always passive, and they are neither easy nor uneasy; so we will assume that the branch nodes are numbered 1 to m in preorder. Let $f_p = p$ for all branch nodes, except when p is a passive uneasy node for which the nearest uneasy node to its right is active; in the latter case, f_p should point to the nearest active uneasy node to its left. (For purposes of this definition, we imagine that artificial nodes 0 and $m+1$ are present at the left and right, both of which are uneasy and active.)

F1. [Initialize.] Set $f_p \leftarrow p$ for $0 \leq p \leq m$; also set $t_0 \leftarrow 1$, $v_0 \leftarrow 0$, and set each z_p so that $r_{z_p} = d_p$.

F2. [Select node p.] Set $q \leftarrow m$; then while $t_q = v_q$ set $q \leftarrow q-1$. Set $p \leftarrow f_q$ and $f_q \leftarrow q$; terminate the algorithm if $p = 0$.

F3. [Change d_p.] Set $s \leftarrow d_p$, $s' \leftarrow r_s$, $k \leftarrow v_p$, and $d_p \leftarrow s'$. (Now $k = v_s \neq v_{s'}$.)

F4. [Update the values.] Set $q \leftarrow s$ and $v_q \leftarrow k \oplus 1$. While $d_q \neq 0$, set $q \leftarrow d_q$ and $v_q \leftarrow k \oplus 1$. (Now q is a leaf that has entered the config if $k = 0$, left it if $k = 1$.) Similarly, set $q \leftarrow s'$ and $v_q \leftarrow k$. While $d_q \neq 0$, set $q \leftarrow d_q$ and $v_q \leftarrow k \oplus 1$. (Now q is a leaf that has left the config if $k = 0$, entered it if $k = 1$.)

F5. [Visit.] Visit the current config, represented by all the leaf values.

F6. [Passivate p?] (All uneasy nodes to p's right are now active.) If $d_p \neq z_p$, return to step F2. Otherwise set $z_p \leftarrow s$, $q \leftarrow p-1$; while $t_q = v_q$, set $q \leftarrow q-1$. (Now q is the first uneasy node to the left of p; we will make p passive.) Set $f_p \leftarrow f_q$, $f_q \leftarrow q$, and return to F2. ▌

Although step F4 may change uneasy nodes to easy nodes and vice versa, the focus pointers need not be updated, because they're still set correctly.

100. A complete program, called GRAYSPSPAN, appears on the author's website. Its asymptotic efficiency can be proved by using the result of exercise 110 below.

102. If so, ordinary spanning trees can be listed in a *strong revolving-door order*, where the edges that enter and leave at each step are adjacent.

Interesting algorithms to generate all the oriented spanning trees with a given root have been developed by Harold N. Gabow and Eugene W. Myers, *SICOMP* **7** (1978), 280–287; S. Kapoor and H. Ramesh, *Algorithmica* **27** (2000), 120–130.

103. (a) Toppling increases $(x_0, x_1, \ldots, x_n)$ lexicographically, but does not change $x_0 + \cdots + x_n$. If we can topple at both V_i and V_j, either order gives the same result.

(b) Adding a grain of sand changes the 16 stable states as follows:

Given	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
+ 0001	0001	0010	0011	0001	0101	0110	0111	0101	1001	1010	1011	1001	1101	1110	1111	1101
+ 0010	0010	0011	0001	0010	0110	0111	0101	0110	1010	1011	1001	1010	1110	1111	1101	1110
+ 0100	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111	0100	0101	0110	0111
+ 1000	1000	1001	1010	1011	1100	1101	1110	1111	0100	0101	0110	0111	1000	1001	1010	1011

The recurrent states are the nine cases with $x_1 + x_2 > 0$ and $x_3 + x_4 > 0$. Notice that repeated addition of 0001 leads to the infinite cycle $0000 \to 0001 \to 0010 \to 0011 \to 0001 \to 0010 \to \cdots$; but the states 0001, 0010, and 0011 are *not* recurrent.

(c) If $x = \sigma(x+t)$ then also $x = \sigma(x+kt)$ for all $k \geq 0$. All components of t are positive; thus $x = \sigma(x + \max(d_1, \ldots, d_n)t)$ is recurrent. Conversely, suppose $x = \sigma(d+y)$, where all $y_i \geq 0$; then $d+y+t$ topples to $x+t$ and it also topples to $\sigma(d)+y+t = d+y$. Therefore $\sigma(x+t) = \sigma(d+y) = x$.

(d) There are $N = \det(a_{ij})$ classes, because elementary row operations (exercise 4.6.1–19) triangularize the matrix while preserving congruence.

(e) There are nonnegative integers $m_1, \ldots, m_n, m'_1, \ldots, m'_n$ such that

$$x + m_1a_1 + \cdots + m_na_n \;=\; x' + m'_1a_1 + \cdots + m'_na_n \;=\; y, \text{ say.}$$

For sufficiently large k, the vector $y+kt$ topples in $m_1 + \cdots + m_n$ steps to $x+kt$, and in $m'_1 + \cdots + m'_n$ steps to $x'+kt$. Therefore $x = \sigma(x+kt) = \sigma(x'+kt) = x'$.

(f) The triangularization in (d) shows that $x \equiv x + Ny$ for arbitrary vectors y. And toppling preserves congruence; hence every class contains a recurrent state.

(g) Since $a = a_1 + \cdots + a_n$ in a balanced digraph, we have $x \equiv x + a$. If x is recurrent, we see in fact that every vertex topples exactly once when $x+a$ reduces to x, because the vectors $\{a_1, \ldots, a_n\}$ are linearly independent.

Conversely, if $\sigma(x+a) = x$ we must prove that x is recurrent. Let $z_m = \sigma(ma)$; there must be some positive k and m with $z_{m+k} = z_m$. Then every vertex topples k times when $z_m + ka$ reduces to z_m; hence there are vectors $y_j = (y_{j1}, \ldots, y_{jn})$ with $y_{jj} \geq d_j$ such that $(m+k)a$ topples to y_j. It follows that $x + n(m+k)a$ topples to $x + y_1 + \cdots + y_n$, and $\sigma(x + y_1 + \cdots + y_n) = \sigma(x + n(m+k)a) = x$.

(h) Treating subscripts cyclically, the spanning trees with arcs $V_j \to V_0$ for $j = i_1$, $\ldots$, i_k have $n-k$ other arcs: $V_j \to V_{j-1}$ for $i_l < j \leq i_l + q_l$ and $V_j \to V_{j+1}$ for

$i_l + q_l < j < i_{l+1}$. The recurrent states, similarly, have $x_j = 2$ for $j = i_1, \ldots, i_k$, and $x_j = 1$ for $i_l < j < i_{l+1}$, except that $x_j = 0$ when $j = i_l + q_l$ and $q_l > 0$.

(i) In this case state $x = (x_1, \ldots, x_n)$ is recurrent if and only if $(n-x_1, \ldots, n-x_n)$ solves the parking problem in the hint, because $t = (1, \ldots, 1)$, and a sequence that doesn't get parked leaves a "hole" that stops $x + t$ from toppling to x.

Notes: This sandpile model, introduced by Deepak Dhar [*Phys. Review Letters* **64** (1990), 1613–1616], has led to many papers in the physics literature. Dhar noted that, if M grains of sand are introduced at random, each recurrent state is equally probable as $M \to \infty$. The present exercise was inspired by the work of R. Cori and D. Rossin, *European J. Combinatorics* **21** (2000), 447–459.

Sandpile theory proves that every digraph D yields an abelian group whose elements correspond somehow to the oriented spanning trees of D with root V_0. In particular, the same is true when D is an ordinary graph, with arcs $u \to v$ and $v \to u$ whenever u and v are adjacent. Thus, for example, we can "add" two spanning trees; and some spanning tree can be regarded as "zero." An elegant correspondence between spanning trees and recurrent states, in the special case when D is an ordinary graph, has been found by R. Cori and Y. Le Borgne, *Advances in Applied Math.* **30** (2003), 44–52. But no simple correspondence is known for general digraphs D. For example, suppose $n = 2$ and $(e_{10}, e_{12}, e_{20}, e_{21}) = (p, q, r, s)$; then there are $pr + ps + qr$ oriented trees, and the recurrent states correspond to generalized two-dimensional toruses as in exercise 7–00. Yet even in the "balanced" case, when $p + q \geq s$ and $r + s \geq q$, no easy mapping between spanning trees and recurrent states is apparent.

104. (a) If $\det(\alpha I - C) = 0$, there is a vector $x = (x_1, \ldots, x_n)^T$ such that $Cx = \alpha x$ and $\max(x_1, \ldots, x_n) = x_m = 1$ for some m. Then $\alpha = \alpha x_m = c_{mm} - \sum_{j \neq m} e_{mj} x_j \geq c_{mm} - \sum_{j \neq m} e_{mj} = 0$. (Incidentally, a real symmetric matrix whose eigenvalues are nonnegative is called *positive semidefinite*. Our proof establishes the well-known fact that any real symmetric matrix with $c_{mm} \geq |\sum_{j \neq m} c_{mj}|$ for $1 \leq m \leq n$ has this property.) Thus $\alpha_0 \geq 0$; and $\alpha_0 = 0$ because $C(1, \ldots, 1)^T = (0, \ldots, 0)^T$.

(b) $\det(xI - C(G)) = x(x-\alpha_1)\ldots(x-\alpha_{n-1})$; and the coefficient of x is $(-1)^{n-1}n$ times the number of spanning trees, by the matrix tree theorem.

(c) $\det(\alpha I - C(K_n)) = \det((\alpha - n)I + J) = (\alpha - n)^{n-1}\alpha$ by exercise 1.2.3–36; here J is the matrix of all 1s. The aspects are therefore $0, n, \ldots, n$.

105. (a) If $e_{ij} = a + be'_{ij}$ we have $C(G) = naI - aJ + bC(G')$. And if C is any matrix whose row sums are zero, the identity

$$\det(xI + yJ - zC) = \frac{x + ny}{x}\, z^n \det((x/z)I - C)$$

can be proved by adding columns 2 through n to column 1, factoring out $(x + ny)/x$, subtracting y/x times column 1 from columns 2 through n, then subtracting columns 2 through n from column 1. Therefore, by setting $x = \alpha - na$, $y = a$, $z = b$, $a = 1$, and $b = -1$, we find that G has the aspects $0, n - \alpha_{n-1}, \ldots, n - \alpha_1$. (In particular, this result agrees with exercise 104(c) when G' is the empty graph $\overline{K_n}$.)

(b) Sort $\{\alpha'_0, \ldots, \alpha'_{n'-1}, \alpha''_0, \ldots, \alpha''_{n''-1}\}$ into order. (An easy case, for variety.)

(c) Here $\overline{G} = \overline{G'} + \overline{G''}$, so G's aspects are $\{0, n' + n'', n'' + \alpha'_1, \ldots, n'' + \alpha'_{n'-1}, n' + \alpha''_1, \ldots, n' + \alpha''_{n''-1}\}$ by (a) and (b). (In particular, G is $K_{m,n}$ when $G' = \overline{K_m}$ and $G'' = \overline{K_n}$, hence the aspects of $K_{m,n}$ are $\{0, (n-1)\cdot m, (m-1)\cdot n, m+n\}$.)

(d) $C(G) = I_{n'} \otimes C(G'') + C(G') \otimes I_{n''}$, where I_n denotes the $n \times n$ identity matrix and $\otimes$ denotes the Kronecker product of matrices. The aspects of $C(G)$ are $\{\alpha'_j + \alpha''_k \mid 0 \leq j < n', 0 \leq k < n''\}$; for if A and B are arbitrary matrices whose eigenvalues

are $\{\lambda_1, \ldots, \lambda_m\}$ and $\{\mu_1, \ldots, \mu_n\}$, respectively, the eigenvalues of $A \otimes I_n + I_m \otimes B$ are the mn sums $\lambda_j + \mu_k$. *Proof:* Choose S and T so that $S^- AS$ and $T^- BT$ are triangular. Then use the matrix identity $(A \otimes B)(C \otimes D) = AC \otimes BD$ to show that $(S \otimes T)^- (A \otimes I_n + I_m \otimes B)(S \otimes T) = (S^- AS) \otimes I_n + I_m \otimes (T^- BT)$. (In particular, repeated use of this formula shows that the aspects of the n-cube are $\{\binom{n}{0}\cdot 0, \binom{n}{1}\cdot 2, \ldots, \binom{n}{n}\cdot 2n\}$, and Eq. (57) follows from exercise 104(b).)

(e) When G is a regular graph of degree d', its aspects are $\alpha_j = d' - \lambda_{j+1}$, where $\lambda_1 \geq \cdots \geq \lambda_n$ are the eigenvalues of the adjacency matrix $A = (e_{ij})$. The adjacency matrix of G' is $A' = B^T B - d' I_{n'}$, where $B = (b_{ij})$ is the $n \times n'$ incidence matrix with entries $b_{ij} =$ [edge i touches vertex j], and where $n = n'd'/2$ is the number of edges. The adjacency matrix of G is $A = BB^T - 2I_n$. Now we have

$$x^n \det(xI_{n'} - B^T B) \;=\; x^{n'} \det(xI_n - BB^T);$$

this identity follows from the fact that the coefficients of $\det(xI - A)$ can be expressed in terms of trace(A^k) for $k = 1, 2, \ldots$, via Newton's identities (exercise 1.2.9–10). So the aspects of G are the same as those of G', plus $n - n'$ aspects equal to $2d'$. [This result is due to E. B. Vakhovsky, *Sibirskiĭ Mat. Zhurnal* **6** (1965), 44–49; see also H. Sachs, *Wissenschaftliche Zeitschrift der Technischen Hochschule Ilmenau* **13** (1967), 405–412.]

(f) $A = A' \otimes A''$, so the aspects are $\{d''\alpha'_j + d'\alpha''_k - \alpha'_j\alpha''_k \mid 0 \leq j < n', 0 \leq k < n''\}$.

(g) $A(G) = I_{n'} \otimes A'' + A' \otimes I_{n''} + A' \otimes A'' = (I_{n'} + A') \otimes (I_{n''} + A'') - I_n$ yields the aspects $\{(d''+1)\alpha'_j + (d'+1)\alpha''_k - \alpha'_j\alpha''_k \mid 0 \leq j < n', 0 \leq k < n''\}$.

106. (a) If α is an aspect of the path P_n, there's a nonzero solution $(x_0, x_1, \ldots, x_{n+1})$ to the equations $\alpha x_k = 2x_k - x_{k-1} - x_{k+1}$ for $1 \leq k \leq n$, with $x_0 = x_1$ and $x_n = x_{n+1}$. If we set $x_k = \cos(2k-1)\theta$, we find $x_0 = x_1$ and $2x_k - x_{k-1} - x_{k+1} = 2x_k - (2\cos 2\theta)x_k$; hence $2 - 2\cos 2\theta = 4\sin^2\theta$ will be an aspect if we choose θ so that $x_n = x_{n+1}$ and so that the x's are not all zero. Thus the aspects of P_n turn out to be $\sigma_{0n}, \ldots, \sigma_{(n-1)n}$.

We must have $\alpha_1 \ldots \alpha_{n-1} = n$, by exercise 104(b), since $c(P_n) = 1$; therefore

$$c(P_m \times P_n) \;=\; \prod_{j=1}^{m-1}\prod_{k=1}^{n-1}(\sigma_{jm} + \sigma_{kn}).$$

(b, c) Similarly, if α is an aspect of the cycle C_n, there's a nonzero solution to the stated equations with $x_n = x_0$. For this case we try $x_k = \cos 2k\theta$ and find solutions when $\theta = j\pi/n$ for $0 \leq j < \lceil n/2 \rceil$. And $x_k = \sin k\theta$ gives further, linearly independent solutions for $\lceil n/2 \rceil \leq j < n$. The aspects of C_n are therefore $\sigma_{0n}, \sigma_{2n}, \ldots, \sigma_{(2n-2)n}$; and we have

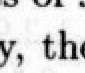
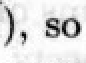
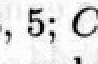
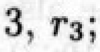
$$c(P_m \times C_n) = n \prod_{j=1}^{m-1}\prod_{k=1}^{n-1}(\sigma_{jm} + \sigma_{(2k)n}), \quad c(C_m \times C_n) = mn \prod_{j=1}^{m-1}\prod_{k=1}^{n-1}(\sigma_{(2j)m} + \sigma_{(2k)n}).$$

Let $f_n(x) = (x + \sigma_{1n}) \ldots (x + \sigma_{(n-1)n})$ and $g_n(x) = (x + \sigma_{2n}) \ldots (x + \sigma_{(2n-2)n})$. These polynomials have integer coefficients; indeed, $f_n(x) = U_{n-1}(x/2+1)$ and $g_n(x) = 2(T_n(x/2+1) - 1)/x$, where $T_n(x)$ and $U_n(x)$ are the Chebyshev polynomials defined by $T_n(\cos\theta) = \cos n\theta$ and $U_n(\cos\theta) = (\sin(n+1)\theta)/\sin\theta$. The calculation of $c(P_m \times P_n)$ can be reduced to the evaluation of an $m \times m$ determinant, because it is the resultant of $f_m(x)$ with $f_n(-x)$; see exercise 4.6.1–12. Similarly, $\frac{1}{n}c(P_m \times C_n)$ and $\frac{1}{mn}c(C_m \times C_n)$ are the respective resultants of $f_m(x)$ with $g_n(-x)$ and of $g_m(x)$ with $g_n(-x)$.

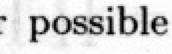
Let $\alpha_n(x) = \prod_{d \backslash n} f_d(x)^{\mu(n/d)}$; thus $\alpha_1(x) = 1$, $\alpha_2(x) = x + 2$, $\alpha_3(x) = (x+3) \times (x+1)$, $\alpha_4(x) = x^2 + 4x + 2$, $\alpha_5(x) = (x^2 + 5x + 5)(x^2 + 3x + 1)$, $\alpha_6(x) = x^2 + 4x + 1$,

etc. By considering so-called field polynomials one can show that $\alpha_n(x)$ is irreducible over the integers when n is even, otherwise it is the product of two irreducible factors of the same degree. Similarly, if $\beta_n(x) = \prod_{d\backslash n} g_d(x)^{\mu(n/d)}$, it turns out that $\beta_n(x)$ is the square of an irreducible polynomial when $n \geq 3$. These facts account for the presence of fairly small prime factors in the results. For example, the largest prime factor in $c(P_m \times P_n)$ for $m \leq n \leq 10$ is 1009; it occurs only in the resultant of $\alpha_6(x)$ with $\alpha_9(-x)$, which is $662913 = 3^2 \cdot 73 \cdot 1009$.

107. There are $(1, 1, 2, 6, 21)$ nonisomorphic graphs for $n = (1, \ldots, 5)$; but we need consider only cases with $\leq \frac{1}{2}\binom{n}{2}$ edges, because of exercise 105(a). The surviving cases when $n = 4$ are free trees: The star is the complement of $K_1 + K_3$, with aspects 0, 1, 1, 4; and P_4 has aspects $0, 2-\sqrt{2}, 2, 2+\sqrt{2}$ by exercise 106. There are three free trees when $n = 5$: The star has aspects 0, 1, 1, 1, 5; P_5's aspects are $0, 2-\phi, 3-\phi, 1+\phi, 2+\phi$; and the aspects of  are $0, r_1, 1, r_2, r_3$, where $(r_1, r_2, r_3) \approx (0.52, 2.31, 4.17)$ are the roots of $x^3 - 7x^2 + 13x - 5 = 0$.

Finally, there are five cases with a single cycle:  is $K_1 \mp (K_2 + \overline{K_2})$, so its aspects are 0, 1, 1, 3, 5; C_5 has aspects $0, 3-\phi, 3-\phi, 2+\phi, 2+\phi$;  has aspects $0, r_1, r_2, 3, r_3$; its complement  has aspects $0, 5-r_3, 2, 5-r_2, 5-r_1$; and the aspects of  turn out to be $0, (5-\sqrt{13})/2, 3-\phi, 2+\phi, (5+\sqrt{13})/2$.

108. Given a digraph D on vertices $\{V_1, \ldots, V_n\}$, let e_{ij} be the number of arcs from V_i to V_j. Define $C(D)$ and its aspects as before. Since $C(D)$ is not necessarily symmetric, the aspects are no longer guaranteed to be real. But if α is an aspect, so is its complex conjugate $\bar{\alpha}$; and if we order the aspects by their real parts, again we find $\alpha_0 = 0$. The formula $c(D) = \alpha_1 \ldots \alpha_{n-1}/n$ remains valid if we now interpret $c(D)$ as the *average* number of *oriented* spanning trees, taken over all n possible roots V_j. The aspects of the transitive tournament T_n, whose arcs are $V_i \to V_j$ for $1 \leq i < j \leq n$, are obviously $0, 1, \ldots, n-1$; and those of its subgraphs are equally obvious.

The derivations in parts (a)–(d) of answer 105 carry over without change. For example, consider $K_1 \bar{+} T_3$, which has aspects 0, 2, 3, 4; this digraph D has $(2, 4, 6, 12)$ oriented spanning trees with the four possible roots, and $c(D)$ is indeed equal to $2\cdot3\cdot4/4$. Notice also that the digraph  is its own complement, and that it has the same aspects as T_3.

Directed graphs also admit another family of interesting operations: If D' and D'' are digraphs on disjoint sets of vertices V' and V'', consider adding a arcs $v' \to v''$ and b arcs $v'' \to v'$ whenever $v' \in V'$ and $v'' \in V''$. By manipulating determinants as in answer 105(a), we can show that the resulting digraph has aspects $\{0, an'' + bn', an'' + \alpha'_1, \ldots, an'' + \alpha'_{n'-1}, bn' + \alpha''_1, \ldots, bn' + \alpha''_{n''-1}\}$. In the special case $a = 1$ and $b = 0$, we can conveniently denote the new digraph by $D' \to D''$; thus, for example, $T_n = K_1 \to T_{n-1}$. The digraph $K_{n_1} \to K_{n_2} \to \cdots \to K_{n_m}$ on $n_1 + n_2 + \cdots + n_m$ vertices has aspects $\{0,\, n_m \cdot s_m,\, \ldots,\, n_2 \cdot s_2,\, (n_1 - 1) \cdot s_1\}$, where $s_k = n_k + \cdots + n_m$.

The aspects of the oriented path Q_n from V_1 to V_n are obviously $0, 1, \ldots, 1$. The oriented cycle O_n has aspects $\{0, 1-\omega, \ldots, 1-\omega^{n-1}\}$, where $\omega = e^{2\pi i/n}$.

There is also a nice result for arc digraphs: The aspects of D^* are obtained from those of D by simply adding $\tau_k - 1$ copies of the number σ_k, for $1 \leq k \leq n$, where τ_k is the in-degree of V_k and σ_k is its out-degree. (If $\tau_k = 0$, we *remove* one aspect equal to σ_k.) The proof is similar to, but simpler than, the derivation in answer 2.3.4.2–21.

Historical remarks: The results in exercises 104(b) and 105(a) are due to A. K. Kelmans, *Avtomatika i Telemekhanika* **26** (1965), 2194–2204; **27**, 2 (February 1966), 56–65; English translation in *Automation and Remote Control* **26** (1965), 2118–2129;

27 (1966), 233–241. Miroslav Fiedler [*Czech. Math. J.* **23** (1973), 298–305] introduced exercise 105(d), and proved interesting results about the aspect α_1, which he called the "algebraic connectivity" of G. Germain Kreweras, in *J. Combinatorial Theory* **B24** (1978), 202–212, enumerated spanning trees on grids, cylinders, and toruses, as well as oriented spanning trees on directed toruses such as $O_m \times O_n$. An excellent survey of graph aspects was published by Bojan Mohar in *Graph Theory, Combinatorics and Applications* (Wiley, 1991), 871–898; *Discrete Math.* **109** (1992), 171–183. For a thorough discussion of important families of graph eigenvalues and their properties, including a comprehensive bibliography, see *Spectra of Graphs* by D. M. Cvetković, M. Doob, and H. Sachs, third edition (1995).

109. Perhaps there is also a sandpile-related reason; see exercise 103.

110. By induction: Suppose there are $k \geq 1$ parallel edges between u and v. Then $c(G) = kc(G_1) + c(G_2)$, where G_1 is G with u and v identified, and G_2 is G with those k edges removed. Let $d_u = k + a$ and $d_v = k + b$.

Case 1: G_2 is connected. Then $ab > 0$, so we can write $a = x + 1$ and $b = y + 1$. We have $c(G_1) > \alpha\sqrt{x+y+1}$ and $c(G_2) > \alpha\sqrt{xy}$, where α is a product over the other $n - 2$ vertices; and it is easy to verify that

$$k\sqrt{x+y+1} + \sqrt{xy} \;\geq\; \sqrt{(x+k)(y+k)}.$$

Case 2: There are no such u and v for which G_2 is connected. Then every multi-edge of G is a bridge; in other words, G is a free tree except for parallel edges. In this case the result is trivial if there's a vertex of degree 1. Otherwise suppose u is an endpoint, with $d_u = k$ edges u — v. If $d_v > k + 1$, we have $c(G) = kc(G_1) > \alpha k\sqrt{x}$ where $d_v = k + 1 + x$, and it is easy to check that $k\sqrt{x} > \sqrt{(k-1)(k+x)}$ when $x > 0$. If $d_v = k$ we have $c(G) = k > \sqrt{(k-1)^2}$. Finally if $d_v = k + 1$, let $v_0 = u$, $v_1 = v$, and consider the unique path $v_1 — v_2 — \cdots — v_r$ where $r > 1$ and v_r has degree greater than 2; only one edge joins v_j to v_{j+1} for $1 \leq j < r$. Again the induction goes through.

[Other lower bounds on the number of spanning trees have been derived by A. V. Kostochka, *Random Structures and Algorithms* **6** (1995), 269–274.]

111. 2 1 5 4 11 7 9 8 6 10 15 12 14 13 3.

112. Either p appears on an even level and is an ancestor of q, or q appears on an odd level and is an ancestor of p.

113. prepostorder(F^R)=postpreorder$(F)^R$ and postpreorder(F^R)=prepostorder$(F)^R$.

114. The most elegant approach, considering that the forest might be empty, is to set things up so that `CHILD(Λ)` points to the root of the leftmost tree, if any. Then initiate the first visit by setting `Q` $\leftarrow \Lambda$, `L` $\leftarrow -1$, and going to step Q6.

115. Suppose there are n_e nodes on even levels and n_o nodes on odd levels, and that n'_e of the even-level nodes are nonleaves. Then steps (Q1, ..., Q7) are performed respectively $(n_e + n_o,\ n_o,\ n'_e,\ n_e,\ n'_e,\ n_o + 1,\ n_e)$ times, including one execution of Q6 because of answer 114.

116. (a) This result follows from Algorithm Q.

(b) In fact, non-ordinary nodes strictly alternate between lucky and unlucky, beginning and ending with a lucky one. *Proof:* Consider the forest F' obtained by deleting the leftmost leaf of F, and use induction on n.

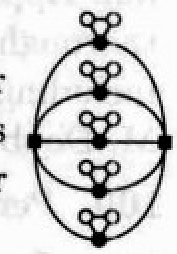
117. Such forests are precisely those whose left-child/right-sibling representation is a degenerate binary tree (exercise 31). So the answer is 2^{n-1}.

118. (a) t^{k-2}, for $k > 1$; luckiness occurs only near extreme leaves.

(b) An interesting recurrence leads to the solution $(F_k + 1 - (k+1) \bmod 3)/2$.

119. Label each node x with the value $v(x) = \sum\{\, 2^k \mid k$ is an arc label on the path from the root to $x\}$. Then the node values in prepostorder are exactly the Gray binary code Γ_n, because exercise 113 shows that they satisfy recurrence 7.2.1.1–5.

(If we apply the same value labeling to the ordinary binomial tree T_n and traverse it in preorder, we simply get the integers $0, 1, \ldots, 2^n - 1$.)



120. False: Only four of the "hollow" vertices in the illustration can appear next to the two "square" vertices, in a Hamiltonian cycle; one hollow pair is therefore out of luck. [See H. Fleischner and H. V. Kronk, *Monatshefte für Mathematik* **76** (1972), 112–117.]

121. Furthermore, there is a Hamiltonian path from u to v in T^2 if and only if similar conditions hold; but we retain u and/or v in $T^{(\prime)}$ if they have degree 1, and we require that the path in (i) be inside the path from u to v (excluding u and v themselves). Condition (ii) is also strengthened by changing 'vertices of degree 4' to 'dangerous vertices', where a vertex of $T^{(\prime)}$ is called dangerous if it either has degree 4 or has degree 2 and equals u or v. The smallest impossible case is $T = P_4$, with u and v chosen to be the non-endpoints. [*Časopis pro Pěstování Matematiky* **89** (1964), 323–338.]

Consequently T^2 contains a Hamiltonian cycle if and only if T is a *caterpillar*, namely a free tree whose derivative is a path. [See Frank Harary and A. J. Schwenk, *Mathematika* **18** (1971), 138–140.]

122. (a) We can represent an expression by a binary tree, with operators at the internal nodes and digits at the external nodes. If binary trees are implemented as in Algorithm B, the essential constraint imposed by the given grammar is that, if $r_j = k > 0$, then the operator at node j is $+$ or $-$ if and only if the operator at node k is $\times$ or $/$. Therefore the total number of possibilities for a tree with n leaves is $2^n S_{n-1}$, where S_n is a Schröder number; namely 10,646,016 when $n = 9$. (See exercise 66, but interchange left with right.) We can rather quickly generate them all, encountering exactly 1640 solutions. Only one expression, namely $1+2/((3-4)/(5+6)-(7-8)/9)$, does the job with no multiplications; twenty of them, such as $(((1-2)/((3/4)\times 5-6))\times 7+8)\times 9$, require five pairs of parentheses; only 15 require no parentheses whatever.

(b) Now there are $1 + \sum_{k=1}^{8} \binom{8}{k} 2^{k+1} S_k = 23{,}463{,}169$ cases, and 3365 solutions. The shortest, of length 12, was found by Dudeney [*The Weekly Dispatch* (18 June 1899)], namely $123 - 45 - 67 + 89$; but he wasn't sure at the time that it was best. The longest solutions have length 27; there are twenty of them, as mentioned above.

(c) The number of cases rises dramatically to $2 + \sum_{k=1}^{8} \binom{8}{k} 4^{k+1} S_k = 8{,}157{,}017{,}474$, and there now are 96,504 solutions. The longest, which is unique, has length 40: $((((.1/(.2+.3))/.4)/.5)/(.6-.7))/(.8-.9)$. There are five amusing examples such as $.1 + (2+3+4+5)\times 6+7+8+.9$, with seven $+$'s; furthermore, there are ten like $(1-.2-.3-4-.5-6)\times(7-8-9)$, with seven $-$'s.

> *There is in fact very little principle in the thing,*
> *and there is no certain way of demonstrating*
> *that we have got the best possible solution.*
> — HENRY E. DUDENEY (1899)

Notes: Marie Leske's *Illustriertes Spielbuch für Mädchen*, first published in 1864, contained the earliest known appearance of such a problem; in the eleventh edition

(1889), the fact that $100 = 1+2+3+4+5+6+7+8\times 9$ was the solution to puzzle 16 in section 553. See also the references in exercise 7.2.1.1–111.

Richard Bellman explained in *AMM* **69** (1962), 640–643, how to handle the special case of part (a) in which the operators are restricted to be either + and ×, without parentheses. His technique of dynamic programming can be used also in this more general problem to reduce the number of cases being considered. The idea is to determine the rational numbers obtainable from every subinterval of the digits $\{1,\ldots,n\}$, having a given operator at the root of the tree. We can also save a good deal of computation by discarding cases for the subintervals $\{1,\ldots,8\}$ and $\{2,\ldots,9\}$ that cannot lead to integer solutions. In this way the number of essentially different trees to consider is reduced to (a) 2,747,275 cases; (b) 6,834,708; (c) 741,167,401.

Floating point arithmetic is unreliable in this application. But the exact rational arithmetic routines of Section 4.5.1 do the job nicely, never needing to work with an integer greater than 10^9 in absolute value.

123. (a) 2284; but $2284 = (1+2\times 3)\times(4+5\times 67)-89$. (b) 6964; but $6964 = (1/.2)\times 34+5+6789$. (c) 14786; but $14786 = -1+2\times(.3+4+5)\times(6+789)$. [If we allow also a minus sign at the left of the expression, as Dudeney did, we actually obtain 1361, 2758, and 85054 additional solutions to problems 120(a), (b), and (c), including nineteen longer expressions in case (a) such as $-(1-2)\times((3+4)\times(5-(6-7)\times 8)+9)$. With such an extension, the smallest unreachable numbers in the present problem become (a) 3802, (b) 8312, and (c) 17722.] The total number of representable integers (positive, negative, or zero) turns out to be (a) 27,666; (b) 136,607; (c) 200,765.

124. Horton–Strahler numbers originated in studies of river flows: R. E. Horton, *Bull. Geol. Soc. Amer.* **56** (1945), 275–370; A. N. Strahler, *Bull. Geol. Soc. Amer.* **63** (1952), 1117–1142. Many tree-drawing ideas are explored and illustrated in a classic paper by Viennot, Eyrolles, Janey, and Arquès, *Computer Graphics* **23**,3 (July 1989), 31–40.

SECTION 7.2.1.7

1. Perhaps under hexagram 21, "crunching" (䷔); however, the ancient commentators related this hexagram more to law enforcement than to the interaction of electrons.

2. (a) For the first nucleotide in the codon, let (T, C, A, G) be respectively represented by (□, □, □, □); represent the second nucleotide, similarly, by (□, □, □, □); represent the third by (□, □, □, □); and superimpose those three representations. Thus, for example, hexagram number 34 is ䷡ = □ + □ + □; it represents the codon TTC, which maps to the amino acid F. Under this correspondence, hexagrams 34 through 54 inclusive map into the respective values (F, G, L, Q, W, D, S, –, P, Y, K, A, I, T, N, H, M, R, V, E, C). Moreover, the three hexagrams that map to '–' are numbers 1, 9, and 41, namely ䷀, ䷈, and ䷨, which mean "creation", "taming," and "removal of excess" in the *I Ching*— all quite appropriate for the notion of completing a protein.

(b) Consider the $\binom{64}{6,6,6,4,4,4,4,4,3,3,2,2,2,2,2,2,2,2,2,1,1} \approx 2.3\times10^{69}$ ways to permute the elements of the $4\times4\times4$ genetic code array. Exactly

$$240288040217578979000399368196455132845166871875018555392000000 \approx 2.4\times10^{63}$$

of them contain at least one run of 21 distinct consecutive elements. [Using the principle of inclusion and exclusion one can show that any multiset $\{(n_1{+}1)\cdot x_1, \ldots, (n_r{+}1)\cdot x_r\}$ with r distinct elements and $n_r = 0$ has exactly

$$(n+1)\binom{n}{n_1,\ldots,n_r}r! - \sum_{k=1}^{r}(n+1-k)k!(r-k)!\,a_k \sum_{\substack{0\le d_1,\ldots,d_r\le 1\\ d_1+\cdots+d_r=k}}\binom{n-k}{n_1-d_1,\ldots,n_r-d_r}$$

such permutations, where $n = n_1 + \cdots + n_r$ and a_k is the number of indecomposable permutations with k elements (exercise 7.2.1.2–100).] Thus only about one out of every million permutations has the stated property.

But there are $4!^3\binom{6}{2,2,2} = 1244160$ ways to represent codons as in part (a), and most of them correspond to different permutations of the amino acids (except for interchanging the representations of T and C in third position).

Empirically, in fact, about 31% of all permutations of the 64 hexagrams turn out to have suitable codon mappings. Thus the construction in part (a) gives no reason to believe that the authors of the *I Ching* anticipated the genetic code in any way.

3. Since $F_{31} - 10^6 = F_{28} + F_{22} + F_{20} + F_{18} + F_{16} + F_{14} + F_9$, the millionth is

⏑⏑⏑⏑⏑⏑⏑–⏑⏑⏑–––––⏑⏑⏑⏑–⏑⏑.

Going the other way is easier: $F_{31} - (F_5 + F_8 + F_{10} + F_{16} + F_{18} + F_{27} + F_{30}) = 314159$.

4. One of the two appearances of במדיר on line 4 should be במידר; this glitch may simply be a typographical error. Similarly, one ירדמב on line 8 should be ירמדב. But the six cases with rightmost letters בר appear twice, in lines 3 and 4, while the cases with rightmost בד are missing. Donnolo himself must be responsible for this flaw.

5. The last one should have been □, not □.

6. The nth value m_n in Mersenne's list agrees with $n!$ only for $1 \le n \le 13$ and $15 \le n \le 38$. Mersenne knew that $14! = 87178291200 \ne m_{14} = 8778291200$, because he inserted the missing '1' in his personal copy of the book (now owned by the Bibliothèque Nationale; a facsimile was published in 1975). But the other errors in his table were not merely typographical, because they propagated into subsequent entries, except in the case of m_{50}: $m_{39} = 39! + 10^{26} - 10^{10}$; $m_{40} = 40m_{39}$; $m_{41} = 41m_{40} - 4\cdot10^{25} - 14\cdot10^{11}$;

$m_n = nm_{n-1}$ for $n = 42, 43, 44, 46, 47, 48, 49, 55, 60$, and 62; $m_{50} = 50m_{49} + 10^{66}$; $m_{51} = 51 \cdot 50 \cdot m_{49}$. When he computed $m_{45} = 9 \cdot 45 \cdot m_{44} - 10^{40} + 10^{29}$, he apparently decided to take a shortcut, because it's easy to multiply by 5 or by 9; but he multiplied *twice* by 9. Most of his errors indicate an unreliable multiplication technique, which may have depended on an abacus: $m_{52} = 52m_{51} + 5 \cdot 10^{56} - 2 \cdot 10^{47} + 10^{34}$; $m_{53} = 53m_{52} - 4 \cdot 10^{29}$; $m_{54} = 54m_{53} + 10^{16}$; $m_{57} = 57m_{56} + 10^{33} + 10^{24}$; $m_{58} = 58m_{57} + 10^{67} - 10^{35} + 10^{32} + 11 \cdot 10^{26}$; $m_{59} = 59m_{58} + 10^{66} + 10^{49} - 10^{28}$; $m_{61} = 61m_{60} - 5 \cdot 10^{81}$; $m_{63} = 63m_{62} + 10^{82} - 10^{74}$; $m_{64} = 64m_{63} + 3 \cdot 10^{81} + 10^{67} + 2 \cdot 10^{38} - 2 \cdot 10^{33} - 10^{23}$.

The remaining case, $m_{56} \approx 10.912m_{55}$ is baffling; it is $\equiv 56m_{55}$ (modulo 10^{17}), but its other digits seem to satisfy neither rhyme nor reason. Can they be easily explained?

Notes: Athanasius Kircher must have copied from Mersenne when he tabulated $n!$ for $1 \le n \le 50$ on page 157 of his *Ars Magna Sciendi* (1669), because he repeated all of Mersenne's mistakes. Kircher did, however, list the values $10m_{14}$, $m_{45}/10$, and $10m_{49}$ instead of m_{14}, m_{45}, and m_{49}; perhaps he was trying to make the sequence grow more steadily. It is not clear who first calculated the correct value of 39!; exercise 1.2.5–4 tells the story of 1000!.

7. The basic permutations are 12345, 13254, 14523, 15432, 12453, 14235, 15324, 13542, 12534, 15243, 13425, 14352. But then we find that all 60 of the even permutations are both alive and dead, because (9) differs by an even permutation from (8). (Moreover, if we somehow repair the case $n = 5$, half of the live permutations for $n = 6$ will turn out to be odd.)

8. For example, we can replace (9) by

$$a_n a_3 \ldots a_{n-1} a_2 a_1,\ a_1 a_{n-1} \ldots a_n a_3 a_2,\ \ldots,\ a_{n-1} a_2 \ldots a_{n-2} a_1 a_n,$$

thus flipping the ends and cyclically shifting the other elements in the permutations of (8). This modification works because all permutations have the correct parity, and because the live and dead ones both have a_1 in every possible position. (We essentially have a dual Sims table for the alternating group, as in Eq. 7.2.1.2–(32); but our elements are named $(n, n-1, \ldots, 1)$ instead of $(0, 1, \ldots, n-1)$.)

A simpler way to generate permutations with the proper signs was published by É. Bézout [*Mémoires Acad. Royale des Sciences* (Paris, 1764), 292]: Each permutation $a_1 \ldots a_{n-1}$ of $\{1, \ldots, n-1\}$ yields n others, $a_1 \ldots a_{n-1} a_n - a_1 \ldots a_{n-2} a_n a_{n-1} + \cdots$.

9. (٠, ١, ٢, ٣, ٤, ٥, ٦, ٧, ٨, ٩); or perhaps we should say (٩, ٨, ٧, ٦, ٥, ٤, ٣, ٢, ١, ٠). *Notes:* A different system was used for the index numbers of the equations; for example, 'ر' stood for 200. Moreover, it should be noted that (11) is actually a transcription of al-Samaw'al's work into *modern* Arabic; Ahmad and Rashed based their work on a 14th-century copy that used similar but older forms of the digits: (٥, ١, ٢, ٣, ۴, ۵, ٦, ٧, ٨, ٩). Al-Samaw'al himself may well have used numerals of an even earlier vintage.

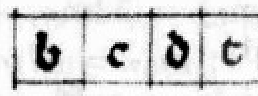
10. If the 56 cases were equally likely, the answer would be $56H_{56} \approx 258.2$, as in the coupon collector's problem (exercise 3.3.2–8). But (6, 30, 20) cases occur with the respective probabilities (1/216, 1/72, 1/36); so the correct answer tu ns out to be

$$\int_0^\infty (1 - (1 - e^{-t/216})^6 (1 - e^{-t/72})^{30} (1 - e^{-t/36})^{20})\, dt \approx 546.6,$$

about 42% of the upper bound $216H_{216}$. [See P. Flajolet, D. Gardy, and L. Thimonier, *Discrete Applied Math.* **39** (1992), 207–229.]

11. It tabulates the $\binom{6}{3} = 20$ combinations of (b, c, d, B, C, D) taken three at a time; furthermore, they appear in lexicographic order if we regard b < c < d < B < C < D.

The letter t (t) means "shift from lowercase to uppercase." [See A. Bonner, *Selected Works of Ramon Llull* (Princeton: 1985), 596–597.] There are two typos: 'd' should be 'b' at the beginning of line 6; 'c' should be 'd' at the end of line 18. Line 1 would have been more consistent with the others if Llull had presented it as

 ;

but in that line, of course, no case shift was needed.

12. Multiply Poinsot's cycle by 5 (mod 7).

13. It's best to have just n lines when there are n different letters:

$$\frac{a.\ aa.\ aaa}{b.\ ab.\ aab.\ aaab.\ bb.\ abb.\ aabb.\ aaabb}$$

Then, assigning the weights $(a, b) = (1, 4)$ gives the numbers 1 through 11 as in (18). (The first line of (16) should also be omitted.) Similarly, for $\{a, a, a, b, b, c\}$ we would implicitly give c the weight 12 and add the additional line

c. ac. aac. aaac. bc. abc. aabc. aaabc. bbc. abbc. aabbc. aaabbc.

[J. Bernoulli *almost* did it right in *Ars Conjectandi*, Part 2, Chapter 6.]

14. ABC ABD ABE ACD ACE ACB ADE ADB ADC AEB AEC AED BCD BCE BCA BDE BDA BDC BEA BEC BED BAC BAD BAE CDE CDA CDB CEA CEB CED CAB CAD CAE CBD CBE CBA DEA DEB DEC DAB DAC DAE DBC DBE DBA DCE DCA DCB EAB EAC EAD EBC EBD EBA ECD ECA ECB EDA EDB EDC. It's a genlex ordering (see Algorithm 7.2.1.3R), proceeding cyclically through the letters not yet used.

[A similar ordering had been used to form all 120 permutations of five letters in a kabbalistic work entitled *Sha'ari Tzedeq*, ascribed to the 13th-century author Natan ben Sa'adyah Har'ar of Messina, Sicily; see *Le Porte della Giustizia* (Milan: Adelphi, 2001).]

15. After j we place the $(n-1)$-combinations of $\{j, \ldots, m\}$ with repetition, so the answer is $\binom{(m+1-j)+(n-1)-1}{n-1} = \binom{m+n-j-1}{n-1}$. [Jean Borrel, also known as Buteonis, pointed this out on pages 305–309 of his early book *Logistica* (Lyon: 1560). He tabulated all throws of n dice for $1 \le n \le 4$, then used a sum over j to deduce that there are $56 + 35 + 20 + 10 + 4 + 1 = 252$ distinct throws for $n = 5$, and 462 for $n = 6$.]

16. N1. [Initialize.] Set $r \leftarrow n$, $t \leftarrow 0$, and $a_0 \leftarrow 0$.

N2. [Advance.] While $r \ge q$, set $t \leftarrow t+1$, $a_t \leftarrow q$, and $r \leftarrow r - q$. Then if $r > 0$, set $t \leftarrow t+1$ and $a_t \leftarrow r$.

N3. [Visit.] Visit the composition $a_1 \ldots a_t$.

N4. [Find j.] Set $j \leftarrow t, t-1, \ldots$, until $a_j \ne 1$. Terminate the algorithm if $j = 0$.

N5. [Decrease a_j.] Set $a_j \leftarrow a_j - 1$, $r \leftarrow t - j + 1$, $t \leftarrow j$; return to N2. ▌

For example, the compositions for $n = 7$ and $q = 3$ are 331, 322, 3211, 313, 3121, 3112, 31111, 232, 2311, 223, 2221, 2212, 22111, 2131, 2122, 21211, 2113, 21121, 21112, 211111, 133, 1321, 1312, 13111, 1231, 1222, 12211, 1213, 12121, 12112, 121111, 1132, 11311, 1123, 11221, 11212, 112111, 11131, 11122, 111211, 11113, 111121, 111112, 1111111.

Nārāyaṇa's sutras 79 and 80 gave essentially this procedure, but with the strings reversed (133, 223, 1123, ...), because he preferred decreasing colex order. [Śārṅgadeva, in *Saṅgītaratnākara* §5.316–375, had previously developed an elaborate theory for the set of all compositions (rhythms) that can be formed from the basic parts $\{1, 2, 4, 6\}$.]

17. The number V_n of visits is $F^{(q)}_{n+q-1} = \Theta(\alpha_q^n)$; see exercise 5.4.2–7. The number X_n of times step N4 tests $a_j = 1$ satisfies $X_n = X_{n-1} + \cdots + X_{n-q} + 1$, and we find

$X_n = V_0 + \cdots + V_n = (qV_n + (q-1)V_{n-1} + \cdots + V_{n-q+1} - 1)/(q-1) = \Theta(V_n)$. The number Y_n of times step N2 sets $a_t \leftarrow q$ satisfies the same recurrence, and we find $Y_n = X_{n-q}$. And the number of times step N2 finds $r = 0$ turns out to be V_{n-q}.

18. It was MDCLXVI in Roman numerals, where M > D > C > L > X > V > I.

19. Lines 329 and 1022. (Puteanus included 139 such verses among his list of 1022.)

20. With 'tria' preceding 'lumina', there are $5! \times 2! \times (11, 12, 12, 16)$ ways having a dactyl in the (1st, 2nd, 3rd, 4th) foot, respectively; with 'lumina' preceding 'tria' there are $5! \times 2! \times (16, 12, 12, 11)$. So the total is 24480. [Leibniz considered this problem near the end of his *Dissertatio de Arte Combinatoria*, and came up with the answer 45870; but his argument was riddled with errors.]

21. (a) The generating function $1/((1 - zu - yu^2)(1 - zv - yv^2)(1 - zw - yw^2))$ is clearly equal to $\sum_{p,q,r,s,t\geq 0} f(p,q,r;s,t)u^p v^q w^r z^s y^t$.

(b) If 'tibi' is $\smile\smile$ and 'Virgo' is $-\,-$, the number is $3!\,3!$ times $\sum_{k=0}^{3}\bigl(f(2k+1, 6-2k, 2; 3, 3) + f(2k, 6-2k, 2; 2, 3)\bigr)$, namely $36((7+7)+(9+5)+(10+5)+(14+7)) = 2304$. Otherwise 'tibi' is $\smile -$, 'Virgo' is $-\smile$, and the number is $2!\,3!$ times $\sum_{k=0}^{3}\bigl(f(2k, 5-2k, 2; 3, 2) + f(2k, 6-2k, 1; 3, 2)\bigr)$, namely $12((7+6)+(5+4)+(4+4)+(0+6)) = 432$.

(c) The fifth foot begins with the second syllable of 'cælo', 'dotes', or 'Virgo'. Hence the additional number is $3!\,3!\sum_{k=0}^{2} f(2k, 5-2k, 2; 3, 2) = 36(7+5+4) = 576$, and the grand total is $2304 + 432 + 576 = 3312$.

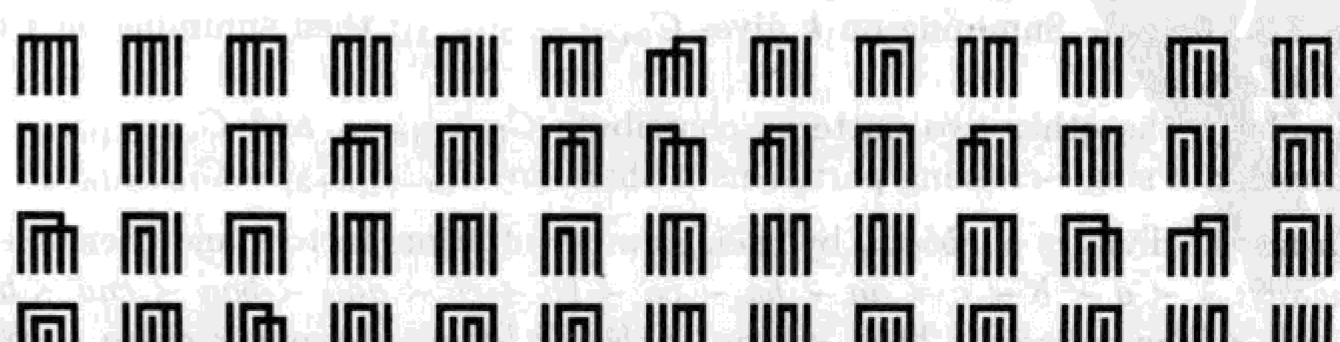
22. Let $\alpha \in \{\text{quot}, \text{sunt}, \text{tot}\}$, $\beta \in \{\text{cælo}, \text{dotes}, \text{Virgo}\}$, $\sigma = \text{sidera}$, and $\tau = \text{tibi}$. Prestet's analysis was essentially equivalent to that of Bernoulli, but he forgot to include the 36 cases $\alpha\alpha\alpha\tau\beta\beta\sigma\beta$. (In his favor one can say that those cases are poetically sterile; Puteanus found no use for them.) The 1675 edition of Prestet's book had also omitted all permutations that end with $\tau\beta$.

Wallis divided the possibilities into 23 types, $T_1 \cup T_2 \cup \cdots \cup T_{23}$. He claimed that his types 6 and 7 each yielded 324 verses; but actually $|T_6| = |T_7| = 252$, because his variable i should be 7, not 9. He also counted many solutions twice: $|T_3 \cap T_5| = 72$, $|T_2 \cap T_7| = |T_5 \cap T_7| = |T_3 \cap T_6| = |T_6 \cap T_{10}| = 36$, and $|T_{11} \cap T_{12}| = |T_{12} \cap T_{13}| = |T_{14} \cap T_{15}| = 12$. He missed the 36 possibilities $\alpha\beta\beta\alpha\sigma\alpha\tau\beta$ (19 of which were used by Puteanus). And he also missed all the permutations of exercise 20(c); Puteanus had used 250 of those 576. The Latin edition of Wallis's book, published in 1693, corrected several typographic errors in this section, but none of the mathematical mistakes.

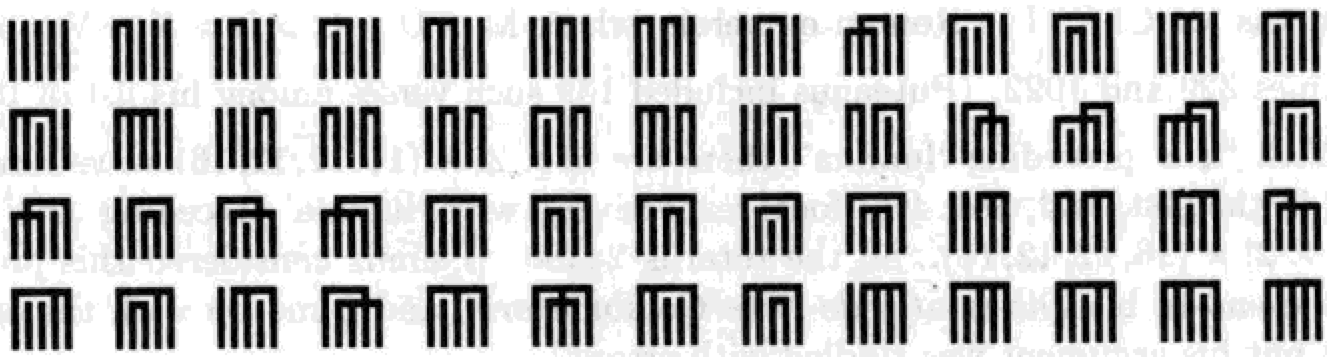
Whitworth and Hartley omitted all cases with 'tibi' $= \smile -$ (see exercise 18), possibly because people's knowledge of classical hexameter was beginning to fade.

[Speaking of errors, Puteanus actually published only 1020 distinct permutations, not 1022, because lines 592 and 593 in his list were identical to lines 601 and 602. But he would have had no trouble finding two more cases — for example, by changing 'tot sunt' to 'sunt tot' in lines 252, 345, 511, 548, 659, 663, 678, 693, or 797.]

23. Reading each diagram left-to-right, so that $12|345 \leftrightarrow$, we get



24. His rule was: For $k = 0, 1, \ldots, n-1$, and for each combination $0 < j_1 < \cdots < j_k < n$ of $n-1$ things taken k at a time, visit all partitions of $\{1, \ldots, n-1\} \setminus \{j_1, \ldots, j_k\}$ together with the block $\{j_1, \ldots, j_k, n\}$. His order for $n = 5$ was:



But strictly speaking, the answer to this exercise is "No" — because Honda's rule is not complete until the order of the combinations is specified. He generated combinations in *colex* order (lexicographic on $j_t \ldots j_1$). Lexicographic order on $j_1 \ldots j_t$ would also be consistent with the list given for $n = 4$, but it would put  before . *Reference:* T. Hayashi, *Tôhoku Math. J.* **33** (1931), 332–337.

25. No; (28) misses 14|235 (the top-bottom reflection of its second pattern).

26. Let a_n be the number of indecomposable partitions of $\{1, \ldots, n\}$, and let a'_n be the number that are both indecomposable and complete. These sequences begin $\langle a_1, a_2, \ldots \rangle = \langle 1, 1, 2, 6, 22, 92, 426, \ldots \rangle$, $\langle a'_1, a'_2, \ldots \rangle = \langle 0, 1, 1, 3, 9, 33, 135, \ldots \rangle$; and the answer to this exercise is $a'_n - 1$ for $n \geq 2$. It turns out that a_n is also the number of symmetric polynomials of degree n in noncommuting variables. [See M. C. Wolf, *Duke Math. J.* **2** (1936), 626–637, who also tabulated indecomposable partitions into k parts.]

If $A(z) = \sum_n a_n z^n$, and if $B(z) = \sum_n \varpi_n z^n$ is the non-exponential generating function for Bell numbers, we have $A(z)B(z) = B(z) - 1$, hence $A(z) = 1 - 1/B(z)$. And the result of exercise 7.2.1.5–35 implies that $\sum_n a'_n z^n = zA(z)/(1 + z - A(z)) = z(B(z)-1)/(1+zB(z))$. Unfortunately $B(z)$ has no especially nice closed form. Notice that indecomposable set partitions with $n > 1$ correspond to vacillating tableau loops with no three consecutive λs equal to zero (see exercise 7.2.1.5–27).

27. The problem is ambiguous because genji-ko diagrams are not well defined. Let's require all vertical lines of a block to have the same height; then, for example, 145|236 has no single-crossing diagram because  is not allowed.

The number of partitions with no crossing is C_n (see exercise 7.2.1.6–26). For one crossing, the elements of the two blocks that cross must appear within the restricted growth sequence as either $x^i y x^j y^k$ or $x^i y^{j+1} x y^k$ or $x^i y^j x y^k x^l$, where $i, j, k, l > 0$.

Suppose the pattern is $x^i y x^j y^k$. There are $t = i + j + k + 2$ "slots" between the $i + 1 + j + k$ elements of this pattern, and the number of ways to fill these slots with noncrossing partitions is $\sum_{i_1 + \cdots + i_t = n-i-j-k-1} C_{i_1} \ldots C_{i_t}$. We can express this number as

$$[z^{n-i-j-k-1}]\, C(z)^{i+j+k+2} = C_{(n-i-j-k-1)n}$$

by Eq. 7.2.1.6–(24). Summing on k gives $C_{(n-i-j-2)(n+1)}$; then summing on j and i gives $C_{(n-4)(n+3)}$.

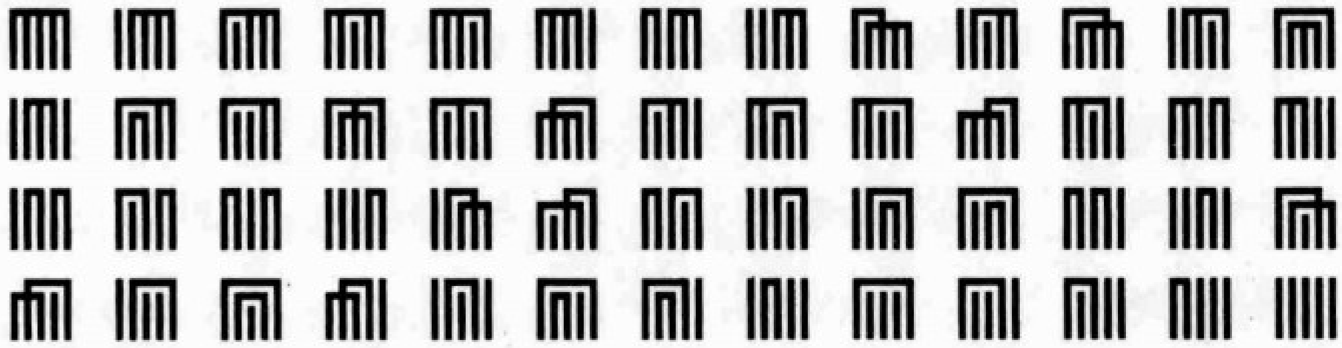
Similarly, the other two patterns contribute $C_{(n-5)(n+3)}$ and $C_{(n-5)(n+4)}$. The total number of single-crossing partitions is therefore $C_{(n-5)(n+3)} + C_{(n-4)(n+4)}$.

28. Order the divisors of *cbbaaa* by their number of prime factors and then colexicographically: $1 \prec a \prec b \prec c \prec aa \prec ba \prec ca \prec bb \prec cb \prec aaa \prec baa \prec caa \prec bba \prec cba \prec cbb \prec baaa \prec caaa \prec bbaa \prec cbaa \prec cbba \prec bbaaa \prec cbaaa \prec cbbaa \prec cbbaaa$.

For every such divisor d, in decreasing order, let d be the first factor; recursively append all factorizations of $cbbaaa/d$ whose first factor is $\preceq d$.

If the divisors had been ordered lexicographically (namely $1 < a < aa < aaa < b < ba < \cdots < cbbaa < cbbaaa$), Wallis's algorithm would have been equivalent to Algorithm 7.2.1.5M with $(n_1, n_2, n_3) = (1, 2, 3)$. He probably chose his more complicated ordering of the divisors because it tends to agree more closely with ordinary numerical order when $a \approx b \approx c$; for example, his ordering is precisely numerical when $(a, b, c) = (7, 11, 13)$. By generating the divisors according to his somewhat complex scheme, Wallis was essentially generating multiset combinations, which we noted in Section 7.2.1.3 are equivalent to bounded compositions. [*Reference: A Discourse of Combinations* (1685), 126–128, with two typographic errors corrected.]

29. The factorizations $edcba$, $edcb\cdot a$, $edca\cdot b$, $\ldots$, $e\cdot d\cdot c\cdot b\cdot a$ correspond respectively to



30. The coefficient is zero unless $i_1 + 2i_2 + \cdots = n$; in that case it is $\binom{m}{k} a_0^{m-k} \binom{k}{i_1, i_2, \ldots}$ where $k = i_1 + i_2 + \cdots$. (Consider $(a_0 z)^m$ times $(1 + (a_1/a_0)z + (a_2/a_0)z^2 + \cdots)^m$.)

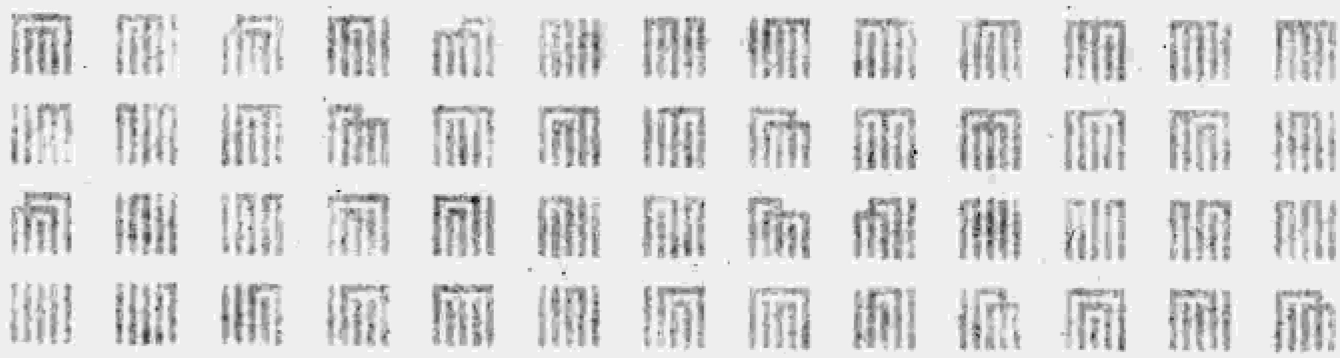
31. The order produced by that algorithm is decreasing lexicographic, the reverse of (31), if we assume that partitions $a_1 \ldots a_k$ have $a_1 \geq \cdots \geq a_k$; de Moivre's was increasing *colexicographic*.

32. $20 \cdot 1 = 7 + 13 \cdot 1 = 2 \cdot 7 + 6 \cdot 1 = 10 + 10 \cdot 1 = 10 + 7 + 3 \cdot 1 = 2 \cdot 10$. In general, Boscovich suggested starting with $n \cdot 1$ and computing the successor of $a \cdot 10 + b \cdot 7 + c \cdot 1$ as follows: If $c \geq 7$, the successor is $a \cdot 10 + (b+1) \cdot 7 + (c-7) \cdot 1$; otherwise if $c + 7b \geq 10$, the successor is $(a+1) \cdot 10 + (c + 7b - 10) \cdot 1$; otherwise stop.

For every such divisor a, in decreasing order, let d be the first factor; recursively append all factorizations of n/a whose first factor is $\le a$.

If the divisors had been ordered lexicographically instead, Wallis's algorithm would have been equivalent to Algorithm 7.2.1.4P with $(n_1, n_2, \ldots)$. He probably chose his more complicated ordering of the divisors because it tends to agree more closely with ordinary numerical order when $a < b < c < d$; for example, his ordering is precisely numerical when $(a, b, c, d) = (2, 7, 11, 13)$. By generating the divisors according to his somewhat complex scheme, Wallis was essentially generating multiset combinations, which we noted in Section 7.2.1.3 are equivalent to bounded compositions. [Reference: *A Discourse of Combinations* (1685), 126–128, with two typographic errors corrected.]

29. The factorizations $dcba$, $dcb\cdot a$, $dca\cdot b$, …, $d\cdot c\cdot b\cdot a$ correspond respectively to



30. The coefficient is zero unless $n_1 + 2n_2 + \cdots = m$; in that case it is $\prod_j a_j^{n_j}\, a_0^{m-k}\binom{k}{n_1, n_2, \ldots}$ where $k = n_1 + n_2 + \cdots$. (Consider $(a_0 z)^m$ times $(1 + (a_1/a_0)z + (a_2/a_0)z^2 + \cdots)^m$.)

31. The order produced by that algorithm is decreasing lexicographic, the reverse of (31), if we assume that partitions $a_1 \ldots a_k$ have $a_1 \ge \cdots \ge a_k$; de Moivre's was increasing colexicographic.

32. $20\cdot 1 = 7 + 13\cdot 1 = 2\cdot 7 + 6\cdot 1 = 10 + 10\cdot 1 = 10 + 7 + 3\cdot 1 = 2\cdot 10$. In general, Boscovich suggested starting with $n\cdot 1$ and computing the successor of $a\cdot 10 + b\cdot 7 + c\cdot 1$ as follows: If $c \ge 7$, the successor is $a\cdot 10 + (b+1)\cdot 7 + (c-7)\cdot 1$; otherwise if $c + 7b \ge 10$, the successor is $(a+1)\cdot 10 + (c + 7b - 10)\cdot 1$; otherwise stop.

计算机程序设计艺术

第4卷　第4册

生成所有树
组合生成的历史

译 者 序

展现在你面前的这一分册是唐纳德 · E克努特（即高德纳）第4卷的第3个分册。此前已经出版了第2和第3分册。

作为计算机科学界的泰斗，高德纳又一次在本分册展现出他严谨的治学态度和追求完美、追求彻底的精神。

本书的正文部分实际上只有两个小节。一是生成所有可能的树。一开头他就指出：我们已经完成了关于经典组合学的概念的研究，即元组、排列、组合和分划。然而，计算机科学家们已经在传统的节目单上加上了称作树的层次体系的另一种基本的模式类。如果不讨论生成所有可能树的问题，显然会感到他的讨论还不够完整。这绝不是他的做事态度。因此，无论如何他都要对这个问题给予回答。

通过把嵌套的括弧和树相联系，使生成所有树的问题同卡塔兰数联系起来。进一步，他又通过把格雷码同树生成及嵌套括弧联系起来。这样一来，看似复杂的生成所有树的问题有了求解的有效途径。我们看到，生成所有树的问题在他手下不再是一个难题了。

使我们印象更深刻的是另一个小节，即关于生成所有可能组合这一问题的历史。在本分册中，高德纳对于几个文明古国以及其他国家在这方面的历史做了透彻的研究和精辟的分析。他首先介绍的是我国。这必然和易经相关，因为易经谈的是阴和阳两方面。这两者自然会有种种组合。然后介绍另一个文明古国印度。古代印度学者从完全不同的方面研究二进制。他们是在词律学中研究二进制，即从抑扬两种音的组合来研究。然后，介绍欧洲包括古希腊和古罗马的诗步。日本在这方面也同样有自己的创造。所以对于日本这一方面的工作，他也做了详细介绍。

我们看到，作为一名严肃的科学家，高德纳在做这些介绍时，采取的是尊重事实、尊重客观的态度，不带任何偏见，保证事实的准确性和调查的彻底性。还需指出，人们或许从某个角度也知道1，2，3，4，5，6，7，8，9这些数字按这个顺序排定，再考虑+，－，*，/ 以及左右括号等组成某个和数。然而，从组合学的角度来论述它们，并把这些事实集中在一起，这是高德纳的一个创造性工作。

生成所有可能性还有一个特殊情况，即括弧的所有可能组合。看看如何组合，可使上述9个顺序的数字及运算符的结果成为100。高德纳给我们介绍了以下的可能性：

$$\begin{aligned}&1+2*3+4*5-6+7+8*9\\&=(1+2-3-4)*(5-6-7-8-9)\\&=100\end{aligned}$$

如果允许多位数字连在一起，我们可以得到：

(1/（2－3+4））*567－89=100

如果还允许使用小数点，还可以得到：

(.1－2－34*.5）/(.6－.789)=100

不仅如此，高德纳还讨论了对于上述三种类型，每种类型共有多少种可能性。比如，对于头一种情况，把这个问题表示为100=（(1/（(2+3）/4－5+6））*7+8）*9的9叉树，并把9个数字做成9个叶，有10 646 016种组合，共有1640个解。其中仅有一种是不使用乘法而只使用+ 、－、/ 的；有两种则需要5对括弧；有15种全然不用括弧。对于第二种情况，共有23 463 169种情况，其中有解的是3365种。对于第三种，则有8 157 017 474种情况，有96 504个解。但是，他没详细介绍结果如何得出。因此，对于读者来说，去算清这些结果仍然是一个激励思考的挑战。

译者

2006年11月于梧州学院

前 言

我喜欢在各种各样的领域中开展工作，使我不会犯明显的错误。

——维克多·克里，(1999)

本册是《计算机程序设计艺术 第4卷 组合算法》的第4册。如第1卷第1册的前言所述，我采用这种分卷分册的出版形式，是因为我知道，完成第4卷的任务将花费许多年，我不能让读者等待太长的时间，而且我希望读者及时提供有价值的反馈意见。

从整卷的情况来看，这一册包括关于组合查找这一冗长章的7.2.1.6节和7.2.1.7节。假设我能维持自己的健康状态，第7章将最终归入三册中（即册4A、4B和4C）。第7章首先概述图论，强调在斯坦福图形库中一些重要图形的某些要点，从这出发我将引出许多例子。然后是7.1节，它涉及按位操作并讨论与布尔函数相关的一些算法。7.2节是关于生成所有可能性的，而且它由7.2.1节“生成基本的组合模式”开始。7.2.1.1~7.2.1.5节讨论生成n元组、排列、组合以及分划的各种方法的细节。下面两节是本册的主要内容，即7.2.1.6节（通过论述怎样生成各种不同类型的树结构而研究基本模式）以及7.2.1.7节（研究前面各节中概念的起源，并介绍与这些概念相关的参考信息）。7.2.2节将概述回溯。如果一切进展顺利，我将继续写下去。目前规划的整个第7章的轮廓，可参见封底所给出的taocp网页。

我怀着无比喜悦的心情来写这部分内容，就像许多年前写第2卷时的感受一样。在写第2卷时我高兴地发现，初等概率论和数论的基本原理很自然地出现在关于随机数生成和算术的算法研究中。而在写作7.2.1节的过程中我了解到，当我们研究组合生成的一些算法时，初等组合学的基本原理也自然地浮现出来，并且具有高度启发性。因此，我可以再次讲述一个美丽的故事。

树结构在所有计算机科学家的心中占据着一个特殊的位置，事实上，很久以来我就想写关于树的生成。尽管我的确很喜欢写7.2.1.1~7.2.1.5节中的元组、排列、组合和分划这些经典的组合结构，但实际上，我更倾心于最后这一部分：宴会结束前的甜品。自1994年以来，我每年在斯坦福大学举办“圣诞树讲座”，谈论该年关于树最值得关注的事实，最后，我把这些讲座的内容汇总成书。这个课题如同甜品一样，内容极为丰富多彩，令人回味无穷。树理论把计算机程序设计中不同方面的许多概念紧密地联结在一起。

7.2.1.7节中关于组合生成的历史，同样令我大脑的另一半得到满足，因为这部分内容涉及诗歌、音乐、宗教、哲学、逻辑以及在世界各地来自不同文化的智力游戏。组合思维的根源很深，我禁不住想，当我把这个故事的各个部分合并在一起时，总的来说，我学习了很多关于人类的事情。

我原来打算简单介绍这些课题，但当我看到，这些思想是何等重要时，我就懂得了，除非我十分透彻地讲解这些基础知识，否则我绝不会感到快乐。因此，我竭尽全力来构筑理论和实践思想的坚实基础，这些思想将支持许多类型的可靠的超级结构。

我要向弗朗克·拉斯基表示感谢，感谢他大胆地向学生们讲授本书的初稿并向我讲述他的授课经验。还有许多其他读者也帮助我检查最初的一些草稿，特别是7.2.1.7节，我经常挑战我的能力极限来理解非英语的内容。

我还将高兴地为头一个向我报告本分册中的每一个错误的人支付2.56美元的酬金，无论这个错误是印刷上的、技术上的还是历史上的。对于我忘了放进索引的那些条目，也有同样的酬金。而对于正文的有价值的改进建议，每一个将获32美分的奖励。（而且，如果你找到一个习题更好的答案，我将通过在最后出版的书中刊登你的名字来给予永久的荣誉，而不仅仅是钱。）

本书对于尚未完成内容的引用，有时以“00”表示，这是以后要予以提供的实际号码的位置。

祝阅读愉快！

D.E.K

加利福尼亚，斯坦福

2005年6月

符号说明。从第7章开始，我将定义关于图的一些运算，来应对当前剧增的许多符号。我将采用如下的一些符号：如果G是关于顶点$U=\{u_1, \cdots, u_m\}$的图，H是关于顶点$V=\{v_1, \cdots, v_n\}$的图，则：

- $G+H$是G与H的和，又叫做连接：它有$m+n$个顶点$U \cup V$和G与H的边。
- $G \mp H$是G与H的协和，又叫做并，即它们的补的连接的补。（因此，它的边是G和H的那些边，加上所有u_j——v_k）。
- $G \times H$是G与H的笛卡儿积：它有mn个顶点$U \times V$，它的边是当u——u'在G中时，为(u, v)——$(u'\ v)$；而当v——v'在H中时，边为(u, v)——(u, v')。
- $G \diamond H$是G和H的直接积，又叫做合取：它的顶点是$U \times V$，但它的边是(u, v)——(u', v')，当且仅当u——u'在G中而v——v'在H中。
- $G \otimes H$是G和H的强积：正如它的符号所示，它组合$G \times H$和$G \diamond H$的边。
- 还有类似于协和的协积。

本册使用而未加说明的其他符号可参见第1、2或3卷中符号索引。那些索引提供了符号使用的地方。当然，第4卷在未来也会有符号索引。



第7章

组合查找

说明：本章的开始部分放在第4卷第0册和第4卷第1册中，这两册计划于2006年和2007年出版（指英文原版。——编辑注）。

7.2 生成所有可能性

7.2.1 生成基本的组合模式

在这一节中，我们的目标是研究在某个组合的空间里遍历所有可能性的方法，因为我们经常面对一些有必要或希望对所有可能性进行穷尽查找的问题。

7.2.1.1 生成所有n元组

我们由简单的开始，考虑如何遍历由n个二进制数字组成的所有2^n个串……

7.2.1.2 生成所有排列

在n元组之后，对于组合生成，几乎每个人都希望列出的下一个最重要的课题的项目是，访问某个给定集合或多重集合的所有排列……

说明：7.2.1.1节和7.2.1.2节的完整内容，可在2005年2月第一次出版的第4卷第2册中找到（中文版已于2006年7月出版，ISBN：7-111-17773-8——编辑注）。

7.2.1.3 生成所有组合

组合数学通常被描述为“关于排列、组合等的研究”，因此，我们现在把注意力转向组合……

7.2.1.4 生成所有分划

理查德·斯坦利的名著*Enumerative Combinatorics*(1986)以讨论十二折方法(The Twelvefold Way）开始，即在实践中经常出现的基本组合问题的$2\times2\times3$数组。我们在这一章前面的各节中已经学习过关于n元组、排列、组合以及复合；我们可以通过学习斯坦利表格中剩下的项目，来完成我们对于经典组合数学的研究，这些项目全部涉及分划。

1 整数的分划是把它写成正整数之和而不考虑顺序的方法。

7.2.1.5 生成所有集合的分划

现在让我们换个话题，关注一个类型不太相同的分划。一个集合的分划是把该集合看作非空的、不相交子集的并的方法。

说明：7.2.1.3节、7.2.1.4节和7.2.1.5节的完整内容，可在2005年7月第一次出版的第4卷第3册中找到（中文版已于2006年7月出版，ISBN：7-111-17774-6——编辑注）。

说明以下序列的重要性：1，2，3，4，5，6，7，8，9，10，…

——理查德·彼·斯坦利，*Enumerative Combinatorics*（1999）

如同一个人的身体有一些成对的成员一样，这些成员有相同的名称，但分为左和右。

当我的讲演以疯狂的思想作为人的身体的一部分，

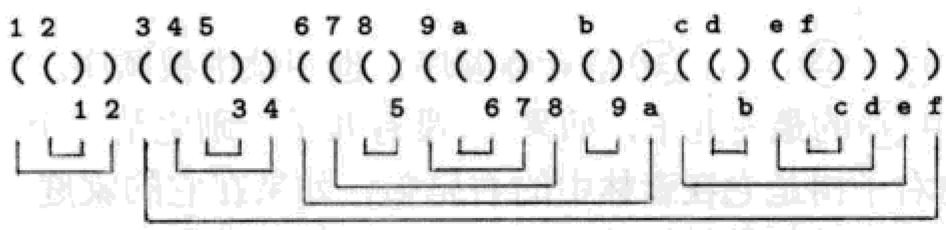
那么，划分为左边部分的讲演被反复地分成越来越小的部分。

——苏格拉底，*Phædrus* 266A (约公元前370年)

7.2.1.6 生成所有树

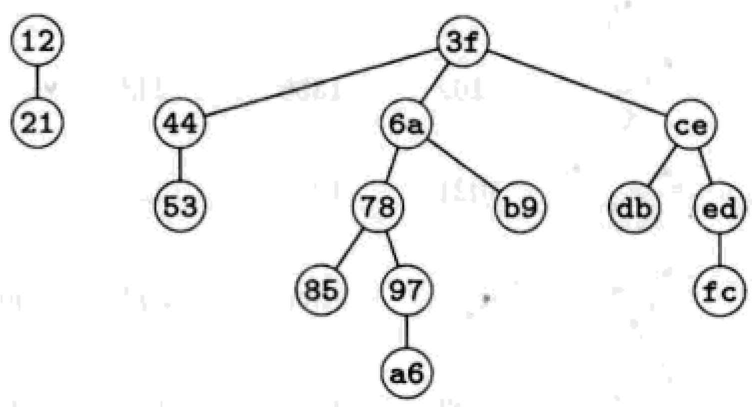
我们已经完成了对经典组合概念：元组、排列、组合以及分划的研究。但是计算机科学家们已经把另一个基本的模式类——称为树的等级结构加到传统的结构中。如同在2.3节和在《计算机程序设计艺术》随后的每一节中都看到的那样，树几乎在计算机科学的每一处茁壮成长。因此，我们来对一些简单算法进行研究。通过这些算法，可以全面地探究各种类型的树。

首先考察一下嵌套括弧和树的森林之间的基本联系。例如：



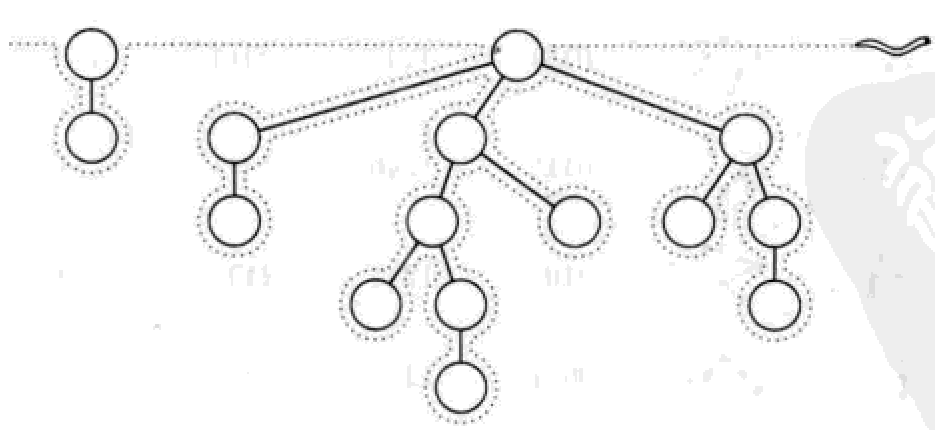
(1) 2

图示包含了标号为1，2，…，f的15个左括弧“(”以及同样标号为1到f的15个右括弧“)”的一个串；串下边的灰线表示这些括弧如何匹配而形成的15对，即12，21，3f，44，53，6a，78，85，97，a6，b9，ce，db，ed和fc。这个串对应于如下所示的森林：



(2)

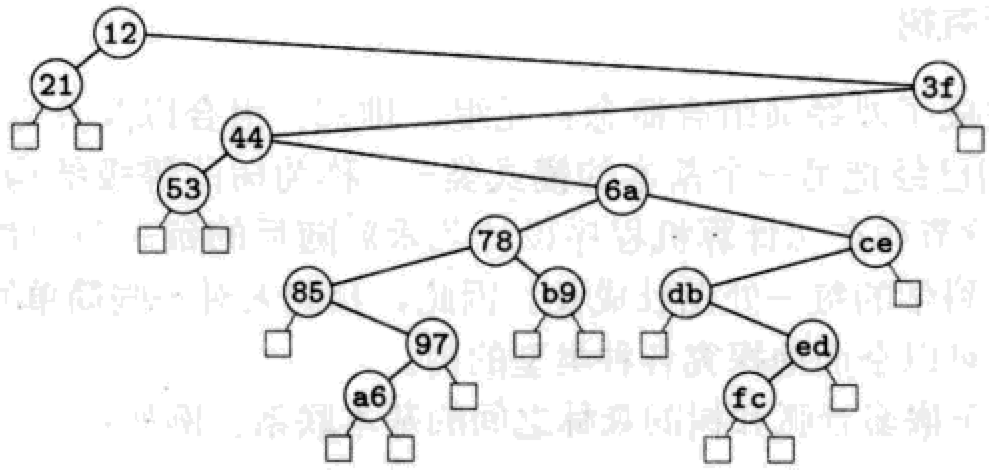
其中节点(12)，(21)，(3f)，...，(fc)是前根顺序（即按第一个坐标排序），节点(21)，(12)，(53)，...，(3f)是后根顺序（即按第二个坐标排序）。如果想像一条蚯蚓沿着森林的外围爬行，



(3)

每当它通过一个节点左边时它看见一个“(”，而当它通过右边时，看见一个“)”，则该蚯蚓将构造出原来的串（1）。

相应地，(2) 中的森林根据2.3.2节中介绍的“自然对应”方法，对应于二叉树



(4)

这里的节点㉑，⑫，53，...，3f是对称顺序，也叫做中根顺序。二叉树中节点(x)的左子树是森林中(x)的最左儿子，如果(x)没有儿子，则它是一个“外部节点”□。在二叉树中(x)的右子树是它在森林中的右兄弟，如果在它的家庭中(x)是最右的儿子，则它是□。森林中诸树的根被认为是兄弟，森林中的最左根是二叉树的根。

3 **表1 当n=4时，嵌套的括弧及相应的对象**

$a_1a_2\ldots a_8$	森林	二叉树	$d_1d_2d_3d_4$	$z_1z_2z_3z_4$	$p_1p_2p_3p_4$	$c_1c_2c_3c_4$	匹配
()()()()			1111	1357	1234	0000	
()()(())			1102	1356	1243	0001	
()(())()			1021	1347	1324	0010	
()(()())			1012	1346	1342	0011	
()((()))			1003	1345	1432	0012	
(())()()			0211	1257	2134	0100	
(())(())			0202	1256	2143	0101	
(()())()			0121	1247	2314	0110	
(()()())			0112	1246	2341	0111	
(()(()))			0103	1245	2431	0112	
((()))()			0031	1237	3214	0120	
((())())			0022	1236	3241	0121	
((()()))			0013	1235	3421	0122	
(((())))			0004	1234	4321	0123	

括弧中的串$a_1a_2\cdots a_{2n}$是正确嵌套的当且仅当它包含n个“(”的出现和n个“)”的出现，其中对于$1\leqslant k\leqslant n$，第k个“(”居于第k个“)”之前。考察所有嵌套的括弧串的最容易的方法是按词典顺序来访问它们。下列算法，把“)”当成是在词典顺序中小于“(”的，它包括由仙波一郎（I.Semba）所提议的某些有效性求精方法。[*Inf. Processing Letters* **12** (1981)，188-192]: 4

算法P（按词典顺序的嵌套括弧）。给定一个整数$n\geqslant 2$，本算法生成嵌套括弧的所有串$a_1a_2\cdots a_{2n}$。

P1. [初始化。]对于$1\leqslant k\leqslant n$，置$a_{2k-1}\leftarrow$ ‘(’ 和$a_{2k}\leftarrow$ ‘)’；也置$a_0\leftarrow$ ‘)’ 和$m\leftarrow 2n-1$。

P2. [访问。]访问嵌套的串$a_1a_2\cdots a_{2n}$（这时，对于$m<k\leqslant 2n$，$a_m=$ ‘(’ $a_k=$ ‘(’）。

P3. [是省事的情况吗？] 置$a_m\leftarrow$ ‘)’。如果$a_{m-1}=$ ‘)’，则置$a_{m-1}\leftarrow$ ‘(’，$m\leftarrow m-1$，并返回P2。

P4. [寻找j。]置$j\leftarrow m-1$和$k\leftarrow 2n-1$。当$a_j=$ ‘(’ 时，置$a_j\leftarrow$ ‘)’，$a_k\leftarrow$ ‘(’，$j\leftarrow j-1$以及$k\leftarrow k-2$。

P5. [增加a_j。] 如果$j=0$则结束本算法。否则，置$a_j\leftarrow$ ‘(’，$m\leftarrow 2n-1$，并返回P2。▌

后边我们将看到，步骤P4中的循环几乎总是短的：平均说来，对于所访问的每个嵌套串，操作$a_j\leftarrow$ ‘)’ 仅执行大约1/3次。

为什么算法P有效呢？令A_{pq}是包含p个左括弧和$q\geqslant p$个右括弧的所有串α的序列，其中$(^{q-p}\alpha$是词典顺序列出的正确嵌套。于是，算法P应当合成A_{nn}，容易看到A_{pq}遵守以下递归规则：

$$A_{pq}=)\,A_{p(q-1)},\ (A_{(p-1)q}\quad \text{如果}0\leqslant p\leqslant q\neq 0;\ A_{00}=\varepsilon \tag{5}$$

如果$p<0$或$p>q$，则A_{pq}为空。A_{pq}的头一个元素是$)^{q-p}(\,)\cdots(\,)$，其中有p对“( )”；最后的元素是$(^{p})^{q}$。因此，词典顺序的生成过程由从右开始扫描，直到找到形如$a_j\cdots a_{2n}=)\,(^{p+1})^{q}$的串，然后，以$(\,)^{q+1-p}(\,)\cdots(\,)$来替换。步骤P4和P5有效地处理这个过程，而步骤P3处理简单情况$p=0$。

表1显示当$n=4$时算法P的输出，以及在(2)和(4)中对应的森林和二叉树。在表1中也列出了其他几个等价的组合对象：例如，一个嵌套的括弧串可以编码为

$$(\,)^{d_1}(\,)^{d_2}\cdots(\,)^{d_n} \tag{6}$$

的游程长度，其中非负整数$d_1d_2\cdots d_n$的约束如下

$$d_1+d_2+\cdots+d_k\leqslant k;\ \text{对于}1\leqslant k<n,\ d_1+d_2+\cdots+d_n=n \tag{7}$$

我们也可以通过序列$z_1z_2\cdots z_n$来表示嵌套的括弧，它确定了左括弧的指数，实质上，$z_1z_2\cdots z_n$是取自集合{1, 2, …, 2n}的n个事物的$\binom{2n}{n}$个组合之一，它的约束如下

$$z_{k-1}<z_k<2k\quad \text{对于}1\leqslant k\leqslant n \tag{8}$$

如果假定$z_0=0$，各个z显然与各个d相关。

$$d_k=z_{k+1}-z_k-1 \quad 对于1\leqslant k<n \tag{9}$$

当把它重写，生成组合$z_1 z_2 \cdots z_n$而不是串$a_1 a_2 \cdots a_{2n}$时，算法P变得特别简单(参见
5 习题2)。

一个括弧串也可通过排列$p_1 p_2 \cdots p_n$表示，其中第k个右括弧同第p_k个左括弧匹配；换句话说，在后根顺序相关联的森林中的第k个节点是前根顺序的第p_k个节点。(由习题2.3.3-20可知，当在后根顺序下对节点进行标号时，节点j是森林中节点k的一个后裔，当且仅当$j<k$和$p_j>p_k$)。通过这样一个规则，即在k的右边恰好有c_k个元素是小于k的（参见习题5.1.1-7）；允许的反演表$c_1 c_2 \cdots c_n$有$c_1=0$，以及

$$0\leqslant c_{k+1}\leqslant c_k+1 \quad 对于\ 1\leqslant k<n \tag{10}$$

而且习题3证明，c_k是在前根顺序森林第k个节点层（第k个左括弧的深度），这是等价于公式

$$c_k=2k-1-z_k \tag{11}$$

表1和习题6也说明了一个特殊类型的匹配：在一个圆桌旁的$2n$个人可以同时互相握手而互不干扰。

因此，算法P确实很有用。如果我们的目标是生成由左链接$l_1 l_2 \cdots l_n$和右链接$r_1 r_2 \cdots r_n$的所有二叉树，则在表1中采用的词典顺序稍显笨拙，我们要从一棵树到它的后继才能得到的数据实际上不易得到。幸而我们有另一个精巧方案：直接生成所有链接二叉树。

算法B（二叉树）。给定$n\geqslant 1$，本算法生成带有n个内节点的所有二叉树，通过左链接$l_1 l_2 \cdots l_n$和右链接$r_1 r_2 \cdots r_n$表示，而且以前根顺序对节点标号。（因此，如节点1总是根，而l_k是k+1或0；如果$l_1=0$且$n>1$，则$r_1=2$）。

B1. [初始化。]对于$1\leqslant k<n$，置$l_k\leftarrow k+1$和$r_k\leftarrow 0$，也置$l_n\leftarrow r_n\leftarrow 0$，并置$l_n\leftarrow r_n\leftarrow 1$（为了在步骤B3处方便起见）。

B2. [访问。]访问由$l_1 l_2 \cdots l_n$和$r_1 r_2 \cdots r_n$所表示的二叉树。

B3. [寻找j。]置$j\leftarrow 1$。当$l_j=0$时，置$r_j\leftarrow 0$，$l_j\leftarrow j+1$且$j\leftarrow j+1$。如果$j>n$，则结束本算法。

B4. [找k和y。]置$y\leftarrow l_j$以及$k\leftarrow 0$。当$r_y>0$时，置$k\leftarrow y$和$y\leftarrow r_y$。

B5. [提高y。]如果$k>0$，则置$r_k\leftarrow 0$，否则置$l_j\leftarrow 0$。然后，置$r_y\leftarrow r_j$，$r_j\leftarrow y$，并返回B2。 ▌

[参看伏·斯卡贝克，*Theoretical Computer Science* **57**(1988)，153-159；步骤B3使用詹·科尔斯的一个观点]。习题44证明，在B3和B4中的两个循环都很短。的确，把一棵链接二叉树转换成它的后继者，平均需要不到9次的内存访问。

表2所示的是当n=4时生成的14棵二叉树，及其对应的森林以及两个相关的序列：数组$e_1 e_2 \cdots e_n$和$s_1 s_2 \cdots s_n$是由在前根顺序下节点k有e_k个孩子以及在相关的森林中有s_k个后裔这一性质定义的。（因此，s_k是二叉树中k的左子树的大小；还有，
6 s_k+1是在2.3.3-（5）介绍的SCOPE链的长度）。下一列在算法P的词典顺序下重复

表1的14个森林，但从左到右镜面反射。而且最后一列显示的是表示协词典顺序的森林的二叉树；它刚好也表示在第4列中的森林，但是通过链接到左兄弟和右子女而不是左子女和右兄弟。最后一列提供了嵌套括弧和二叉树之间相关的联系，因此，它展示了算法B之所以正确的某些道理（参见习题19）。

表2 当n=4时，链接二叉树及相应的对象

$l_1l_2l_3l_4$	$r_1r_2r_3r_4$	二叉树	森林	$e_1e_2e_3e_4$	$s_1s_2s_3s_4$	协词典顺序的森林	左兄弟/右子女
2340	0000			1110	3210		
0340	2000			0110	0210		
2040	0300			2010	3010		
2040	3000			1010	1010		
0040	2300			0010	0010		
2300	0040			1200	3200		
0300	2040			0200	0200		
2300	0400			2100	3100		
2300	4000			1100	2100		
0300	2400			0100	0100		
2000	0340			3000	3000		
2000	4300			2000	2000		
2000	3040			1000	1000		
0000	2340			0000	0000		






• **树的格雷码**。我们先前关于其他组合模式的经验表明，通过只做少量的修改从一个实例到另一个实例，大概就可生成括弧和树。确实，至少有三种非常好的方法可以实现这一目标。

首先考虑嵌套括弧的情况，可以用满足条件(8)的序列$z_1\, z_2 \cdots z_n$来表示。根据7.2.1.3节，我们可以用一个近乎完美的方法生成所有这样的组合，方法是：以某个变量z_j每步变化±1或±2方式遍历所有可能性，在第j个左括弧的邻域中，通过简单地把( )↔)(或(( ))↔))(进行改变，得到每一个括弧串到其后继者的组合。以下是当n=4时的方法：

1357, 1356, 1346, 1345, 1347, 1247, 1245, 1246, 1236, 1234, 1235, 1237, 1257, 1256。我们通过取每一个模式$z_1 z_2 \cdots\ z_{n-1}$并令z_n使用*结束次序*或者像在7.2.1.3-(45)中的它的逆遍历它所有的合法值，可以把对于$n-1$的任何解推广到对于n的一个解。而且从$2n-2$往下进行，而后向上到$2n-1$，或者反过来，并省略$\leqslant z_{n-1}$的所有元素。

算法N（*近乎完美的嵌套括弧*）。这个算法访问$\{1,\cdots,2n\}$的所有n元组合$z_1\cdots z_n$，n表示在一个嵌套串中左括弧的指数，一次只改变一个指数。这个过程由表示临时目标的一个辅助数组$g_1\cdots g_n$加以控制。

N1. [初始化。] 对于$1\leqslant j\leqslant n$，置$z_j\leftarrow 2j-1$，$g_j\leftarrow 2j-2$。

N2. [访问。] 访问n元组合$z_1\cdots\ z_n$。然后置$j\leftarrow n$。

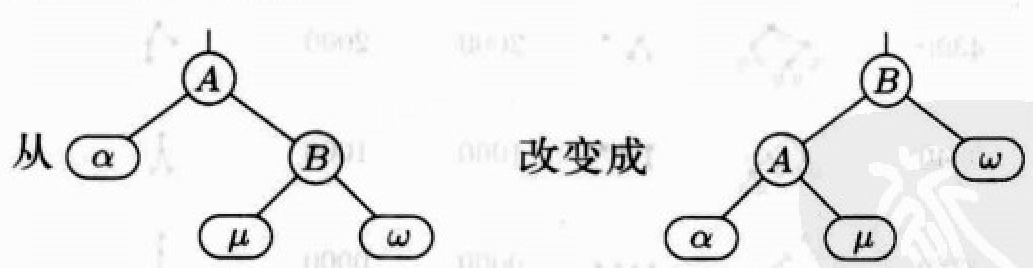
N3. [寻找j。] 如果$z_j=g_j$，则置$g_j\leftarrow g_j\oplus 1$（由此使最低有效位取补），$j\leftarrow j-1$，并重复这一步。

N4. [最后一部分？] 如果g_j-z_j为偶数，则$z_j\leftarrow z_j+2$并返回N2。

N5. [减少或转向。] 置$t\leftarrow z_j-2$。如果$t<0$，则结束此算法。否则，如果$t\leqslant z_{j-1}$，则置$t\leftarrow t+2[t<z_{j-1}]+1$。最后置$z_j\leftarrow t$并且返回到N2。 ▌

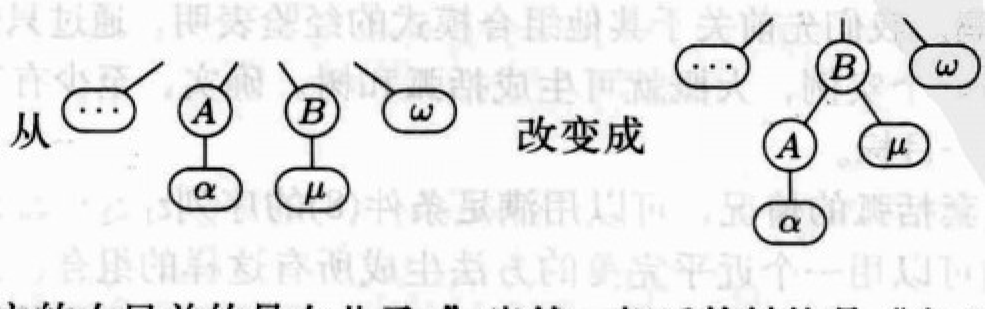
[多·鲁兰特兹·范·巴龙奈吉恩在*J. Algorithms* **35** (2000)，100-107上介绍了一个类似的算法。也可参见相利民、牛岛和夫、唐常杰的文章，*Inf. Proc.Letters* **76** (2000)，169-174。弗·拉斯基和安·普罗斯库洛夫斯基在*J. Algorithms* **11** (1990)，68-84，以前已经证明，当$n\geqslant 4$是偶数时，对于所有表$z_1\cdots z_n$，如何构造*完美的*格雷码，因此在每步中仅仅对某个z_j修改± 1；但构造十分复杂，就实用性而言还没有足够简单的完美方案。习题48证明当$n\geqslant 5$是奇数时，完美性是不可能的。]

如果我们的目的是生成链接树结构而不是括弧串，z下标变化的完美性就嫌不够，因为像( )↔) (这样的简单交换并不一定对应于简单的链接操作。有一个好得
8 多的方法是建立在“转动”算法基础上，该算法能够保持在6.2.3节中的查找树的平衡，*向左转动*把一棵二叉树



(12)

7 这样，对应的森林就



(13)

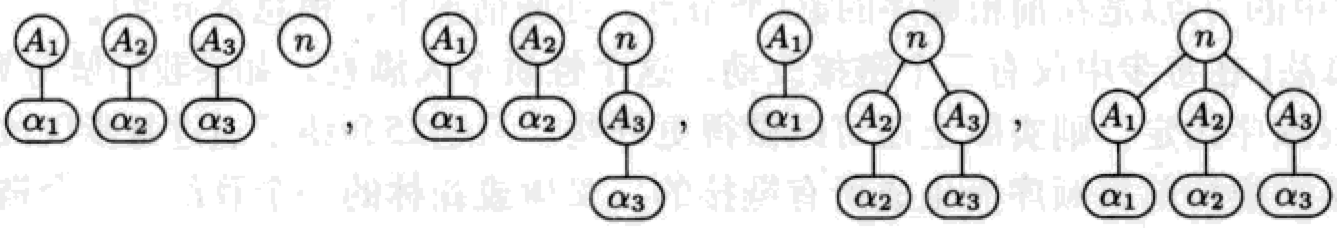
“节点Ⓐ变成它的右兄弟的最左儿子。”当然，相反的转换是“*向右转动*”：“Ⓑ的最左儿子变成它的左兄弟”。(12)中的垂直线表示同整个上下文结构的联系，或者是

一个左链接，或者是一个右链接，或者是根的指针。任意的或所有子树α、μ或ω都可以为空。(13)中的“…”表示在含有$\textcircled{B}$的家庭中左边的其他兄弟，也可以为空。

转动的优点在于仅仅三个链接改变：$\textcircled{A}$的右链接、$\textcircled{B}$的左链接和上边的指针。转动维持二叉树的中根顺序和森林的后根顺序。（还应注意，转动二叉树符合常见的代数公式中的结合律）

$$(\alpha\mu)\omega = \alpha(\mu\omega) \tag{14}$$

一个非常类似n元组的经典反射格雷码的简单方案（算法7.2.1.1H）和排列的平易改动的方法（算法7.2.1.2P）可通过转动生成所有二叉树或森林。考虑$n-1$节点的森林，有k个根$\textcircled{A_1}$，…，$\textcircled{A_k}$。在n个节点有k+1个森林，在前$n-1$个节点有相同的后根顺序序列，但是节点$\textcircled{n}$在最后；例如，当k=3时，通过逐次地把$\textcircled{A_3}$、$\textcircled{A_2}$和$\textcircled{A_1}$向左转动得到的



而且，当$\textcircled{n}$处于右边或者处于顶部的极端情况下，我们对其他$n-1$个节点可以实施希望的转动，因为节点$\textcircled{n}$不在其中。因此，如同由约·马·卢卡斯、多·鲁兰特兹·范·巴龙奈吉恩以及弗·拉斯基所发现的，[*J. Algorithms* **15** (1993), 343-366]，通过简单地令节点$\textcircled{n}$前后漫游，我们可以把任何$(n-1)$节点的树列表推广 9
成为所有n节点的树列表。对于低层细节的特别关注，事实上有可能以显著效率做好这一工作！

算法L（通过转动链接的二叉树）。本算法生成表示n节点二叉树的左链接和右链接的所有数组对$l_0\, l_1 \cdots l_n$和$r_1\, r_2 \cdots r_n$，其中l_0是树的根，而链接(l_k, r_k)分别指向在对称顺序下的第k个节点的左子树和右子树。等于算法L生成所有n节点森林，其中l_k和r_k表示后根顺序第k个节点的左儿子和右兄弟。每个树都通过对它前驱的一个转动得到。通常使用两个辅助的数组$k_1 \cdots k_n$和$o_0\, o_1 \cdots o_n$表示后指针和方向来控制这个过程。

L1. [初始化。] 对于$1 \leqslant j<n$，置$l_j \leftarrow 0$，$r_j \leftarrow j+1$，$k_j \leftarrow j-1$和$o_j \leftarrow -1$，并置$l_0 \leftarrow o_0 \leftarrow 1$，$l_n \leftarrow r_n \leftarrow 0$，$k_n \leftarrow n-1$以及$o_n \leftarrow -1$。

L2. [访问。] 访问$l_0\, l_1 \cdots l_n$和$r_1 \cdots r_n$表示的二叉树或森林。设$j \leftarrow n$，$p \leftarrow 0$。

L3. [寻找j。] 如果$o_j>0$，则置$m \leftarrow l_j$，如果$m \neq 0$，则转向L5。如果$o_j<0$，则置$m \leftarrow k_j$；如果$m \neq 0$，则转向L4；否则置$p \leftarrow j$。无论在哪种情况下，如果m=0，则置$o_j \leftarrow -o_j$，$j \leftarrow j-1$，并重复本步骤。

L4. [向左转动。] 置$r_m \leftarrow l_j$，$l_j \leftarrow m$，$x \leftarrow k_m$和$k_j \leftarrow x$。如果x=0，则置$l_p \leftarrow j$；否则置$r_x \leftarrow j$。返回到L2。

L5. [向右转动。] 如果j=0则终止。否则置$l_j \leftarrow r_m$，$r_m \leftarrow j$，$k_j \leftarrow m$，$x \leftarrow k_m$。如果x=0，则置$l_p \leftarrow m$；否则置$r_x \leftarrow m$，返回到L2。 ▌

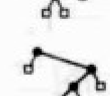
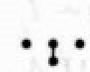
习题38证明，算法L对于生成的每棵树仅需大约9次内存访问，因此它几乎和算法B一样快。(事实上，通过把三个量o_n、l_n和k_n保存到寄存器中，则每一步可节省两次存储。当然算法B也可以加速)。

表3示出当n=4时由算法L访问的二叉树和森林的序列，还有某些辅助表进一步展现这一过程。当排列$q_1\,q_2\,q_3\,q_4$已经以森林的后根顺序(二叉树的对称顺序)加以编号时，它以前根顺序列出节点。它是表1中的排列$p_1\,p_2\,p_3\,p_4$的逆，“协森林”是森林的共轭（左右对称反射）；类似于表2中的$s_1\,s_2\,s_3\,s_4$，数$u_1\,u_2\,u_3\,u_4$是它的范围坐标。最后一列示出所谓的“对偶森林”。在习题11-13、19、24、26和27中剖析了这些相关联的量的重要性。

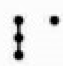
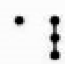
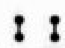
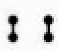
在算法L和表3中的链接$l_0\,l_1\cdots\,l_n$和$r_1\cdots\,r_n$与在算法B和表2中的链接$l_1\cdots\,l_n$和$r_1\cdots\,r_n$是不可比较的，因为算法L维持中根顺序/后根顺序而算法B维持前根顺序。在算法L中的节点k是二叉树中从左到右的第k个节点，所以需要l_0来标识根；但在算法B中的节点k是在前根顺序的第k个节点，在此情况下，根总是节点1。

10 算法L在每步中仅有三个链接改动，这个性质令人满意，如果我们坚持算法B的前根顺序约定，则实际上还可以做得更好些。习题25介绍了通过每步仅改变两个链接来维持前根顺序而生成所有链接的二叉树或森林的一个算法。一个链接变成零，而另一个链接变成非零。这个“剪接-嫁接”算法是前面介绍过的三个“非常好的树的格雷码”的第三个，它只有一个缺点：即它的控制机制比算法L的控制机制稍微复杂点。所以，当我们把判定在每步中要改变什么链接的开销包括在内时，为进行这些计算，大约需要增加40%的时间。

表3 当n=4时通过转动生成的二叉树和森林

$l_0l_1l_2l_3l_4$	$r_1r_2r_3r_4$	$k_1k_2k_3k_4$	二叉树	森林	$q_1q_2q_3q_4$	协森林	$u_1u_2u_3u_4$	对偶
10000	2340	0123			1234		0000	
10003	2400	0122			1243		1000	
10002	4300	0121			1423		2000	
40001	2300	0120			4123		3000	
40021	3000	0110			4132		3100	
10023	4000	0111			1432		2100	
10020	3040	0113			1324		0100	
30010	2040	0103			3124		0200	
40013	2000	0100			4312		3200	

（续）

$l_0l_1l_2l_3l_4$	$r_1r_2r_3r_4$	$k_1k_2k_3k_4$	二叉树	森林	$q_1q_2q_3q_4$	协森林	$u_1u_2u_3u_4$	对偶
40123	0000	0000			4321		3210	
30120	0040	0003			3214		0210	
20100	0340	0023			2134		0010	
20103	0400	0022			2143		1010	
40102	0300	0020			4213		3010	

11

树的棵数。对于由算法P、B、N和L所生成的输出总数，有一个简单的公式，即

$$C_n = \frac{1}{n+1}\binom{2n}{n} = \binom{2n}{n} - \binom{2n}{n-1} \tag{15}$$

我们在等式2.3.4.4-(14)中证明了这一事实。前几项的值是

$n=$	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$C_n=$	1	1	2	5	14	42	132	429	1430	4862	16796	58786	208012	742900

它们称作卡塔兰数，因为由尤金·卡塔兰所写的某些有影响的论文[*Journal de math.* **3** (1838)，508-515；4 (1839)，95-99]而来，斯特林的近似公式告诉我们其渐近值，

$$C_n = \frac{4^n}{\sqrt{\pi} n^{3/2}}\left(1 - \frac{9}{8n} + \frac{145}{128n^2} - \frac{1155}{1024n^3} + \frac{36939}{32768n^4} + O(n^{-5})\right) \tag{16}$$

特别是，我们得到结论

$$\frac{C_{n-k}}{C_n} = \frac{1}{4^k}\left(1 + \frac{3k}{2n} + O\left(\frac{k^2}{n^2}\right)\right) \quad 当|k| \leqslant \frac{n}{2} \tag{17}$$

(当然，由式(15)，C_{n-1}/C_n 精确地等于$(n+1)/(4n-2)$。在2.3.4.4节，我们也推导出生成函数

$$C(z) = C_0 + C_1 z + C_2 z^2 + C_3 z^3 + \cdots = \frac{1 - \sqrt{1-4z}}{2z} \tag{18}$$

并且证明了重要的公式

$$[z^n]C(z)^r = \frac{r}{n+r}\binom{2n+r-1}{n} = \binom{2n+r-1}{n} - \binom{2n+r-1}{n-1} \tag{19}$$

参见习题2.3.4.4-33的答案以及具体的数学等式(5.70)。

这些事实给了我们分析算法P多于足够的信息，该算法按词典顺序生成嵌套括弧。步骤P2显然执行C_n次；步骤P3通常只作简单的改动而后又返回步骤P2。我们需要如何继续到步骤P4呢？容易，就是步骤P2找到$m=2n-1$的次数，m是最右“(”

的位置，所以我们恰好在C_{n-1}个情况下有$m=2n-1$。因此，P3置$m \leftarrow m-1$并立即返回P2的概率，由(17)，是$(C_n-C_{n-1})/C_n \approx 3/4$。另一方面，如果我们走到P4步，我们需要在该步中恰好有$h-1$次$a_j \leftarrow$ ‘(’ 和$a_k \leftarrow$ ‘(’。$h>x$的情况数为以x个平凡的对( )…( )结束的长度为$2n$的嵌套串的个数，即C_{n-x}。因此，在步骤P4中算法改动a_j和a_k的总次数，由(17)是

$$\begin{aligned} C_{n-1}+C_{n-2}+\cdots+C_1 &= C_n\left(\frac{C_{n-1}}{C_n}+\frac{C_{n-2}}{C_n}+\cdots+\frac{C_1}{C_n}\right) \\ &= \frac{1}{3}C_n\left(1+\frac{2}{n}+O\left(\frac{1}{n^2}\right)\right) \end{aligned} \tag{20}$$

12 我们已经证明了前面关于有效性的断言。

为了加深理解，研究奠定算法P的基础的递归结构如同在(5)中所表达的，是有帮助的。在该公式中的序列A_{pq}有C_{pq}个元素，其中

$$C_{pq}=C_{p(q-1)}+C_{(p-1)q}，\text{如果}0 \leqslant p \leqslant q \neq 0;\quad C_{00}=1 \tag{21}$$

如果$p<0$或$p<q$，则$C_{pq}=0$。因此，可以形成三角形数组

$$\begin{matrix} C_{00} \\ C_{01} & C_{11} \\ C_{02} & C_{12} & C_{22} \\ C_{03} & C_{13} & C_{23} & C_{33} \\ C_{04} & C_{14} & C_{24} & C_{34} & C_{44} \\ C_{05} & C_{15} & C_{25} & C_{35} & C_{45} & C_{55} \\ C_{06} & C_{16} & C_{26} & C_{36} & C_{46} & C_{56} & C_{66} \end{matrix} \quad = \quad \begin{matrix} 1 \\ 1 & 1 \\ 1 & 2 & 2 \\ 1 & 3 & 5 & 5 \\ 1 & 4 & 9 & 14 & 14 \\ 1 & 5 & 14 & 28 & 42 & 42 \\ 1 & 6 & 20 & 48 & 90 & 132 & 132 \end{matrix} \tag{22}$$

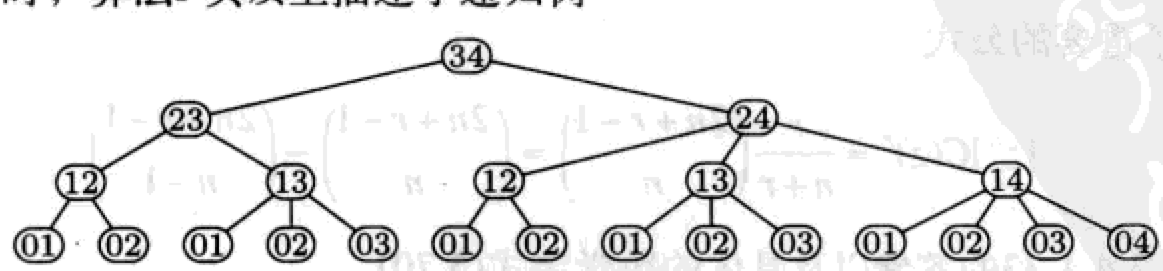
其中每个元素都是它上边最靠近的邻居和左邻居之和；卡塔兰数$C_n=C_{nn}$出现在对角线上。这个三角阵的元素可追溯到1711年德·摩伊弗热的古老谱系，所谓“选票数”，因为它们表示$p+q$张选票的序列，表决不利于获得p张选票而对手有q张选票的候选人。一般公式

$$C_{pq}=\frac{q-p+1}{q+1}\binom{p+q}{p}=\binom{p+q}{p}-\binom{p+q}{p-1} \tag{23}$$

可以通过归纳法或者以各种各样更有趣的方法来证明，参见习题39和习题2.2.1-4的答案。注意，由于(19)，我们有

$$C_{pq}=[z^p]\ C(z)^{p-q+1}. \tag{24}$$

当$n=4$时，算法P实质上描述了递归树



(25)

因为说明(5)意味着

$$A_{nn}=(A_{(n-1)n}\ \text{和}\ A_{pq}=)^{q-p}(A_{(p-1)p,})^{q-p-1}(A_{(p-1)(p+1),})^{q-p-2}(A_{(p-1)(p+2),}\cdots,(A_{(p-1)q}\quad \text{当}0 \leqslant p<q \tag{26}$$

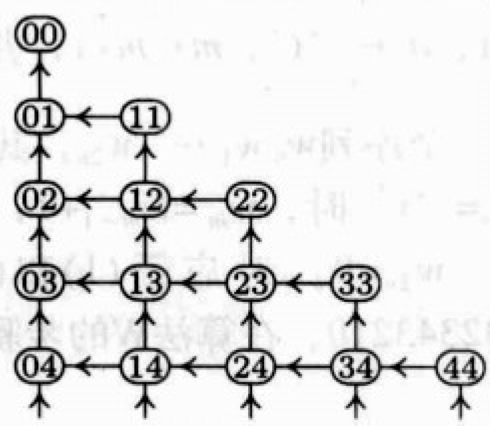
在这个递归树中在节点$\textcircled{pq}$下边的叶数是C_{pq}，而在层$n-1-p$上节点$\textcircled{pq}$恰出现$C_{(n-q)(n-1-p)}$次。因此，我们必定有

$$\sum_q C_{(n-q)(n-1-p)} C_{pq} = C_n \text{，对于} 0 \leqslant p < n \tag{27}$$

(25)中从左到右的14个叶对应于表1从上到下的14行。注意，该表的列$c_1\ c_2\ c_3\ c_4$中的项分别把数0000，0001，0010，…，0123赋给(25)中的叶，这同树节点的“德 13
威（Dewey）十进制记号”相一致（但下标从0开始而不是从1开始，而且有一个额外的0添加在开始处）。

从一片叶爬到另一片叶的蚯蚓沿着递归树的下部，在坐标$c_1 \cdots c_n$的h被改动时，即算法P重置h个‘(’和h个‘)’值时将上升或下降h层，这一发现容易理解前边的结论，即在完整的爬行期间，条件$h>x$恰好出现C_{n-x}次。

当我们思考由递归式(21)所提议的无穷有向图时，出现了理解算法P的另一个方法：



(28)

显然C_{pq}是有向图中从$\textcircled{pq}$到$\textcircled{00}$的路径数。这是由(21)所致。其实，A_{pq}中的每个括弧串直接对应一条以‘(’表明向左一步和以“)”表明向上一步的通路。当扩展一条部分路径时首先尝试向上，算法P系统地剖析所有这样的路径。

因此，从节点$\textcircled{nn}$开始，在节点$\textcircled{pq}$时作如下计算：如果$p=q=0$，则停止，否则，如果$N \leqslant C_{p(q-1)}$，发射‘)’，置$q \leftarrow q-1$，并继续；否则，置$N \leftarrow N - C_{p(q-1)}$，发射‘(’，置$p \leftarrow p-1$，并继续，这样，容易确定算法P所访问的嵌套括弧的第N个串。下列算法[弗朗克·拉斯基博士论文，(加里福尼亚大学、圣地亚哥分校，1978)，16-24]通过跟踪计算C_{pq}来避免对于卡塔兰三角的预先计算需要。

算法U（*去掉嵌套括弧的串的排阶*）。给定n和N，其中$1 \leqslant N \leqslant C_n$，这个算法计算算法P的第$N$个输出$a_1 \cdots a_{2n}$。

U1. [初始化。] 置$q \leftarrow n$和$m \leftarrow p \leftarrow c \leftarrow 1$。当$p<n$时，置$p \leftarrow p+1$和$c \leftarrow ((4p-2)c)/(p+1)$。

U2. [完成了吗？] 如果$q=0$，则终止这个算法。

U3. [向上吗？] 置$c' \leftarrow ((q+1)(q-p)c)/((q+p)(q-p+1))$。(这时我们有$1 \leqslant N \leqslant c = C_{pq}$和$c' = C_{p(q-1)}$。)如果$N \leqslant c'$，则置$q \leftarrow q-1$，$c \leftarrow c'$，$a_m \leftarrow$ ‘)’，$m \leftarrow m+1$，并返回U2。

U4. [向左。] 置$p \leftarrow p-1$，$c \leftarrow c - c'$，$N \leftarrow N - c'$，$a_m \leftarrow$ ‘(’，$m \leftarrow m+1$，并返回U3。 ▌

14 **随机树**。通过简单地将算法U应用于1和C_n之间的随机整数N，我们可以随机地选择嵌套括弧的一个串$a_1 a_2 \cdots a_{2n}$。当n大于32左右时，因为C_n可能十分大，这个思想实际上并非很好。由戴·布·阿尔诺德和迈·罗·斯利普[*ACM Trans. Prog. Languages and Systems* 2(1980)，122-128]所提出的更简单和更好的方法是通过在(28)中在ⓝⓝ处开始，反复地以适当概率取向左或向上的转移来生成随机的“蚯蚓步行”。得到的算法几乎和算法U一样，但它只处理小于n^2+n+1的非负整数。

算法W（嵌套括弧的一致随机的串）。这个算法生成适当嵌套的“(”和“)”的一个随机串$a_1 a_2 \cdots a_{2n}$。

W1. [初始化。] 置$p \leftarrow q \leftarrow n$和$m \leftarrow 1$。

W2. [完成了吗？] 如果$q=0$，则结束。

W3. [向上？] 令X是在范围$0 \leqslant X<(q+p)(q-p+1)$内的一个随机整数。如果$X<(q+1)(q-p)$，则置$q \leftarrow q-1$，$a_m \leftarrow$ ‘)’，$m \leftarrow m+1$，并返回W2。

W4. [向左。] 置$p \leftarrow p-1$，$a_m \leftarrow$ ‘(’，$m \leftarrow m+1$，并返回W3。 ▌

一条蚯蚓步行可以当作一个序列$w_0 w_1 \cdots w_{2n}$，其中w_m是在m步之后蚯蚓当前的深度。因此，$w_0=0$。当$a_m=$ ‘(’ 时，$w_m=w_{m-1}+1$；当$a_m=$ ‘)’ 时，$w_m=w_{m-1}-1$；而且，我们有$w_m \geqslant 0$，$w_{2n}=0$。对应于(1)和(2)的序列$w_0 w_1 \cdots w_{30}$是0121012321234345432321232343210。在算法W的步骤W3处，我们有$q+p=2n+1-m$和$q-p=w_{m-1}$。

我们说一个森林的*轮廓*是对于$0 \leqslant m \leqslant 2n$，遍历平面中的点$(m, -w_m)$的路径，其中$w_0 w_1 \cdots w_{2n}$是对应于嵌套括弧的相关联串$a_1 \cdots a_{2n}$的蚯蚓步行。图36表示，如果画出所有50个节点的森林轮廓并且按照属于它上边的森林的个数，把每一个点涂黑，那会发生的情况。例如，w_1总是1，因此，图36左上方的三角形区域是浓黑的。但w_2为0或为2；0出现在$C_{49} \approx C_{50}/4$的情况中。所以，相邻的菱形区域是75%灰色阴影的。图36示出一个随机森林的形状，这类似于我们在7.2.1.4节和7.2.1.5节的图30、图31和图35中看见的随机分划的形状。

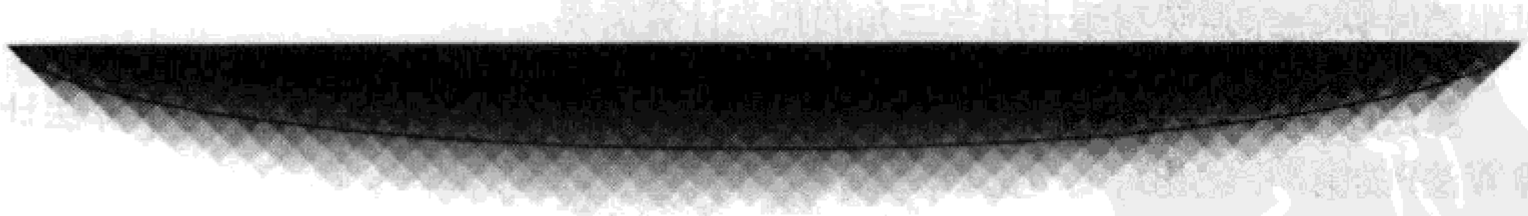


图36 一个随机的50个节点森林的形状

当然，我们不能真实地画出所有这些森林的轮廓，因为它们总共有C_{50}=1,978,261,657,756,160,653,623,774,456个。借助于数学，我们假设已经完成它

15 们了。$w_{2m}=2k$的概率是$C_{(m-k)(m+k)} C_{(n-m-k)(n-m+k)}/C_n$，因为以$m+k$个(和$m-k$个)开始共有$C_{(m-k)(m+k)}$种方法，而以$n-(m+k)$个(和$n-(m-k)$个)结束有$C_{(n-m-k)(n-m+k)}$个方法。由(23)和斯特林近似公式，这个概率是，当对于$0<\theta<1$，$m=\theta n$ 和$n \to \infty$时

$$\frac{(2k+1)^2(n+1)}{(m+k+1)(n-m+k+1)}\binom{2m}{m-k}\binom{2n-2m}{n-m+k}\Big/\binom{2n}{n}$$
$$=\frac{(2k+1)^2}{\sqrt{\pi}(\theta(1-\theta)n)^{3/2}}e^{-k^2/(\theta(1-\theta)n)}\left(1+O\left(\frac{k+1}{n}\right)+O\left(\frac{k^3}{n^2}\right)\right) \tag{29}$$

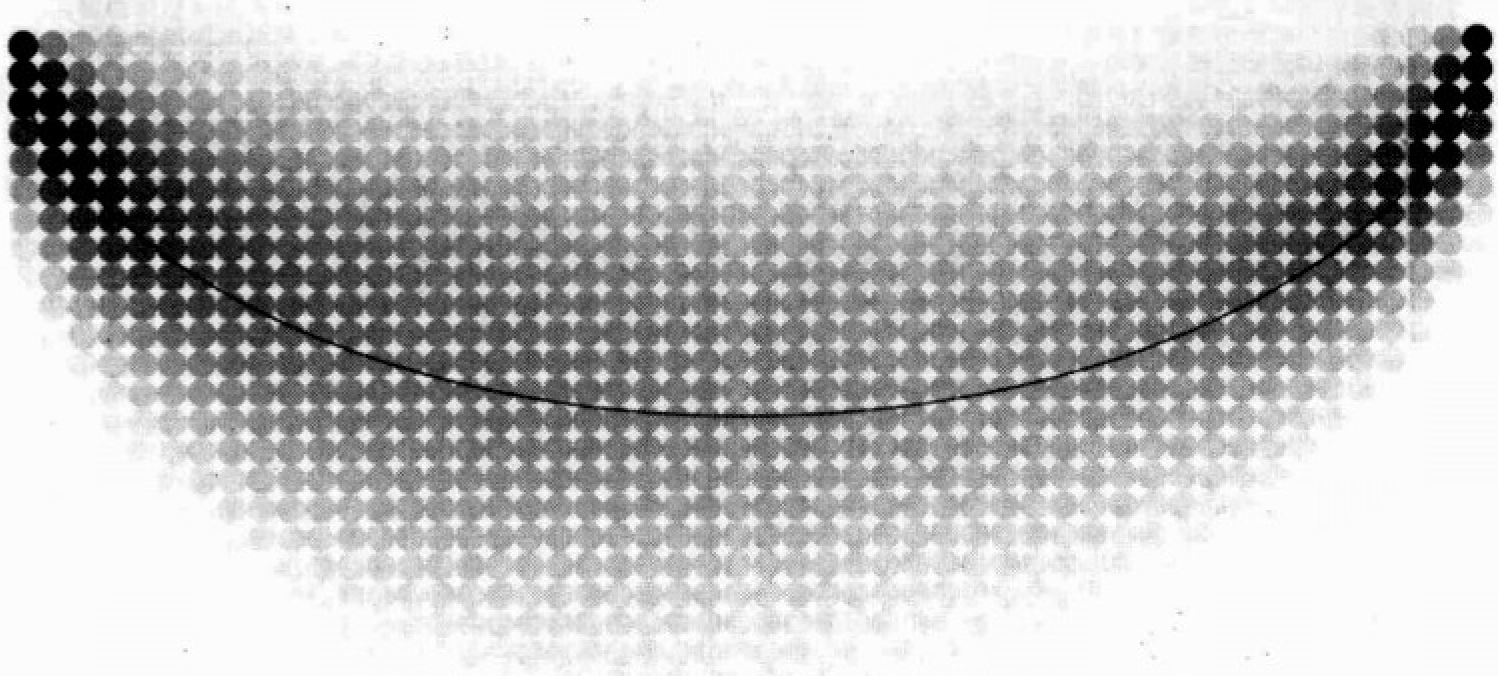


图37 在一个随机50个节点的二叉树中内部节点的位置

在习题57中已经算出w_{2m}的平均值，它是

$$\frac{(4m(n-m)+n)\binom{2m}{m}\binom{2n-2m}{n-m}}{n\binom{2n}{n}}-1=4\sqrt{\frac{\theta(1-\theta)n}{\pi}}-1+O(n^{-1/2}) \tag{30}$$

而对于n=50它图示为图36中的一条曲线。

当n很大时，蚯蚓步行接近于所谓的“布朗旅行”，这是概率论中的一个重要概念。例如，参见保罗·列维，*Processus Stochastiques et Mouvement Brownien* (1948)，225-237；盖伊·劳查德，*J. Applied Prob.* **21**(1984)，479-499以及*BIT* **26**(1986)，17-34；戴维·阿尔度斯，*Electronic Communications in Probability* **3**(1998)，79-90；琼·瓦廉，*Electronic Communications in Probability* **4**(1999)，25-29；吉·弗·玛克特，*Random Structures and Algorithms* **24**(2004)，118-132。

随机二叉树的形状是怎样的？弗朗克·拉斯基在*SIAM J. Algebraic and Discrete Methods* **1**(1980)，43-50中研究了这个问题，而结果答案很有趣。假设我们像在(4)中那样画一棵二叉树，当节点以对称顺序来编号时，第m个内部 16
节点在水平位置m处。如果以这种方法来画所有50个节点的二叉树，并且使它们彼此迭加在一起的话，我们就得到图37中所示的节点位置的分布。类似地，如果以对称顺序从0到n对外部节点编号，把它们放在水平位置0.5，1.5，…，

n+0.5，则所有50个节点的二叉树的“边缘”形成图38所示的分布。注意在极左或极右，根节点最可能是数1或数n；在中间，它有很小可能为$\lfloor (n+1)/2 \rfloor$或$\lceil (n+1)/2 \rceil$。

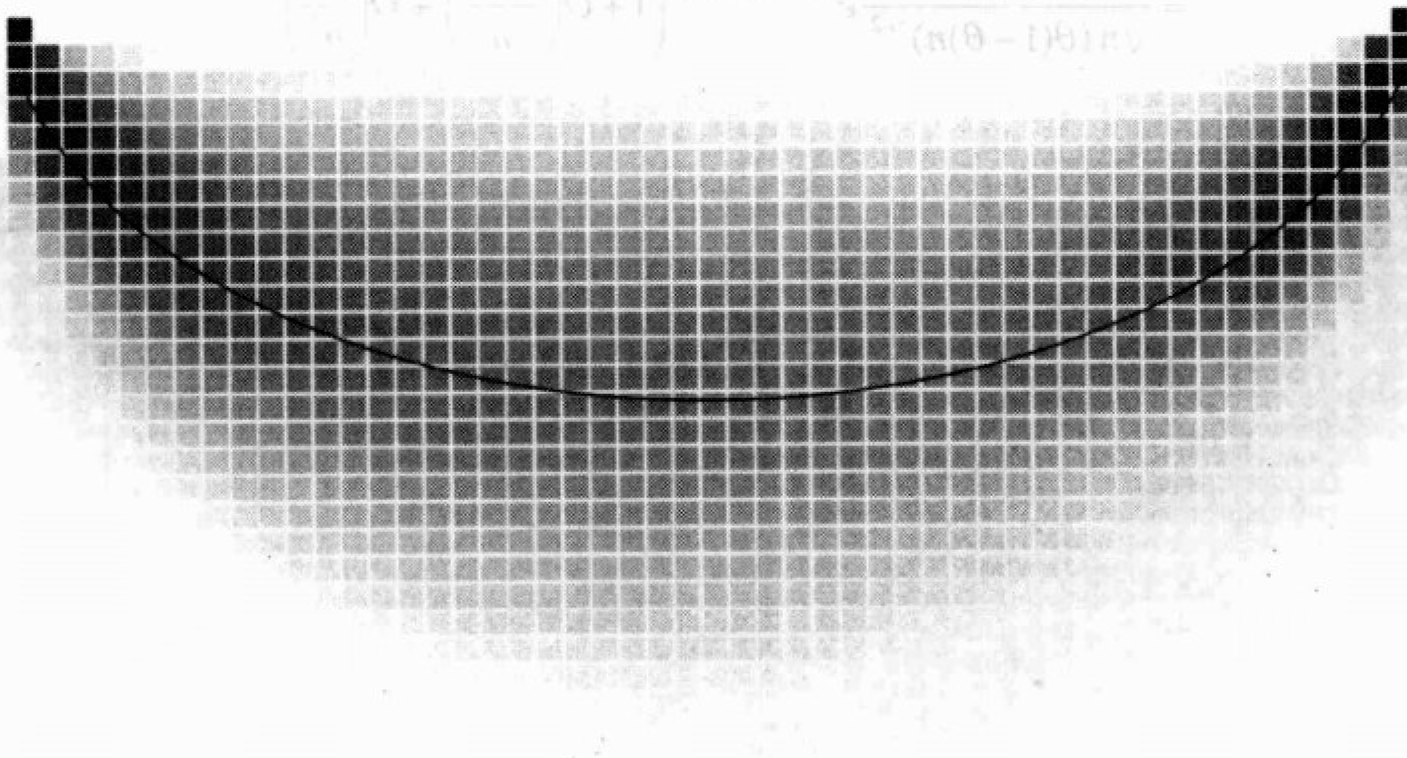


图38 在随机50个节点的二叉树中外部节点的位置

如图36中一样，图37和图38中的光滑曲线表示节点的平均深度；在习题58和习题59导出精确的公式。对于所有满足$0<\theta<1$的固定比率θ，奇怪地像(30)一样，当$m=\theta n$和$n\to\infty$时，外部节点m的平均深度为

$$8\sqrt{\frac{\theta(1-\theta)n}{\pi}}-1+O\left(\frac{1}{\sqrt{n}}\right) \tag{31}$$

而内部节点m的平均深度渐近相同，只是以“-3”代替“-1”。因此，我们可以说，一棵随机二叉树的平均形状近似地是一个椭圆的下半部，n个单位宽和$4\sqrt{n/\pi}$层深。

在习题60、习题61和习题62中讨论了生成森林的随机编码的三种值得注意的方法。它们不如算法W那样直接，但它们有相当大的组合兴趣。头一个以含n个(和m个)的任意随机串开始，它们不必是嵌套的；$\binom{2n}{n}$中的每一个都是同等可能的。
17 然后，把每一个这样的串转换成适当嵌套的一个序列，以这样一个方式进行，即恰好n+1串映射到每一个最终的结果。第二个方法是类似的，但是它以n+1个0和n个2的序列开始，以这样一个方式来映射它们，即恰好2n+1个原来的串产生每一个可能的结果。而第三个方法从刚好含$n-1$个1和n+1个0的n个二进位串产生每个输出。换句话说，这三个方法都提供了C_n同时等于$\binom{2n}{n}\Big/(n+1)$，$\binom{2n+1}{n}\Big/(2n+1)$，和$\binom{2n}{n-1}\Big/n$的事实的组合证明。例如，当$n$=4时，我们有14=70/5=126/9=56/4。

如果要直接以链接形式生成随机二叉树，可以使用由吉-拉·勒密所提议的漂

亮方法[*RAIRO Informatique Théorique* **19**(1985)，179-195]。他的方法特别有教益，因为它示出随机卡塔兰树如何可以真正“自然地”出现，并且使用基于峨林德·罗德里格斯的经典思想的极为简单的机制[*J. de Math.* **3**(1838)，549]。我们假设，目标不仅是得到一棵通常的n个节点的二叉树，而是要得到一棵已修饰过的二叉树，即一棵扩充的二叉树。其中外部节点以某种顺序从0到n作了标号。有$(n+1)!$种方法对给定的二叉树进行修饰，所以，具有n个内部节点的修饰过的二叉树的总数是

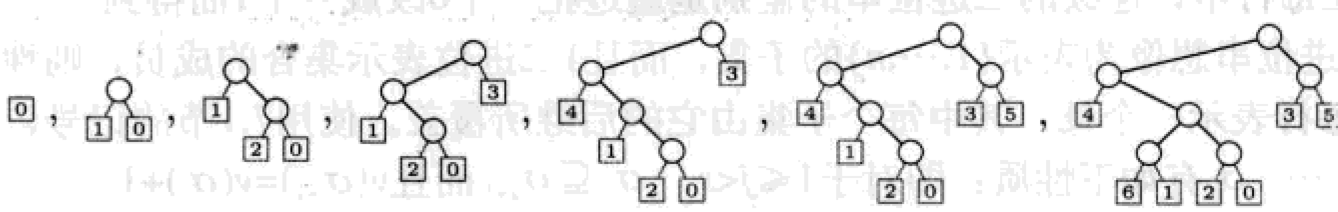
$$D_n=(n+1)!C_n=\frac{(2n)!}{n!}=(4n-2)D_{n-1} \qquad (32)$$

勒密发现，从$n-1$阶给定的修饰过的树来构造一棵n阶的修饰过的树有$4n-2$个容易的方法。我们简单地在给定的树中选择$2n-1$个节点(内部或外部)中的任何一个，比如说是x，并且以


(33)

来代替，于是，插入一个新的内部节点和一片新的叶，同时把x和它的后裔(如果有的话)向下移动一层。

例如，以下是构造6阶的修饰过的树的一个方法。


(34)

注意，每个修饰过的树都是以恰好一种方法通过这一过程得到的。因为每棵树的前驱必定是通过勾销掉最高编号的叶得到的。因此，勒密的构造产生一致地随机的修饰过的树；如果忽略外部节点，我们就得到通常的未加修饰过的品种的随机二叉树。

实现勒密过程的一个吸引人的方法是维持一个链接$L_0\,L_1\cdots\;L_{2n}$的表，其中外部（叶）节点有偶数的号而内部（分支）节点有奇数的号。根是节点L_0；对于$1\leqslant k\leqslant n$，分支节点$2k-1$的左和右儿子分别为L_{2k-1}和L_{2k}，则程序是短的和美妙的。 18

算法R（随机二叉树增长）。这个算法使用上面说明的约定，构造具有N个内部节点的一致随机的二叉树的链接表示$L_0\,L_1\cdots\;L_{2N}$。

R1. [初始化。] 置$n\leftarrow 0$和$L_0\leftarrow 0$。

R2. [完成了吗？]（这时链接$L_0\,L_1\cdots\;L_{2n}$表示一棵随机的n节点二叉树。）如果$n=N$，则算法终止。

R3. [增加n。] 令X是0和$4n+1$之间（含）的一个随机整数。置$n\leftarrow n+1$，$b\leftarrow X \bmod 2$，$k\leftarrow\lfloor X/2\rfloor$，$L_{2n-b}\leftarrow 2n$，$L_{2n-1+b}\leftarrow L_k$，$L_k\leftarrow 2n-1$，并返回到R2。▮

• **子集的链**。既然我们已经牢固地记住树和括弧，现在是讨论圣诞树模式[⊖]的

⊖ 这一名称是由于感伤而选择的，因为这个模式有很像节日树的一般形状，还因为它是作者2002年12月在斯坦福大学第9年度的“圣诞树讲座”的题目。

好时候。圣诞树模式是把所有长度为n的2^n个二进位串排成$\binom{n}{\lfloor n/2\rfloor}$行和$n+1$列的一个著名方法。它是由德·布鲁因、范·厄布本霍斯特·登贝尔根以及克鲁斯维依克发现的[*Nieuw Archief voor Wiskunde* (2), **23**(1951), 191-193]。

1阶的圣诞树模式是单个行‘0 1’，而2阶的模式是

$$\begin{array}{c} \quad 10 \\ 00\ \ 01\ \ 11 \end{array} \tag{35}$$

一般，通过取阶n模式的每行‘$\sigma_1 \quad \sigma_2 \ \cdots \quad \sigma_s$’并以两行

$$\begin{array}{c} \sigma_2 0 \cdots \sigma_s 0 \\ \sigma_1 0 \ \ \sigma_1 1 \ \cdots \ \sigma_{s-1} 1 \ \ \sigma_s 1 \end{array} \tag{36}$$

代替它，就得到阶为$n+1$的圣诞树模式。（当$s=1$时，这些行的头一个省略。）

以这种方式进行，我们就得到出现在表4的阶为8的模式的例子，通过归纳法容易验证

i) 2^n个二进位串的每一个在模式中恰出现一次。

ii) 带有k个1的二进位串全都出现在同一列中。

iii) 在每行中，连续的二进位串的差别是通过把一个0改成一个1而得到。

如果把二进位串想像为表示$\{1,\cdots,n\}$的子集，而且1二进位表示集合的成员，则性质(iii)说，每行表示一个链，其中每个子集由它的后继所覆盖。使用7.1节的记号，每行$\sigma_1 \quad \sigma_2 \quad \cdots \quad \sigma_s$有如下性质：即对于$1\leqslant j<s$，$\sigma_j \subseteq \sigma_{j+1}$而且$\nu(\sigma_{j+1})=\nu(\sigma_j)+1$。

性质(i)和(ii)告诉我们，如果对列编号成由0到n，则在列k中恰有$\binom{n}{k}$个元素。这个发现连同每行都在各列居中这一事实，证明了断言，即总的行数为$\max_{0\leqslant k\leqslant n}$

19 $\binom{n}{k}=\binom{n}{\lfloor n/2\rfloor}$。我们称这个数为$M_n$。

表4　8阶的圣诞树模式

			10101010			
		10101000	10101001	10101011		
			10101100			
		10100100	10100101	10101101		
		10100010	10100110	10101110		
	10100000	10100001	10100011	10100111	10101111	
			10110010			
		10110000	10110001	10110011		
			10110100			
		10010100	10010101	10110101		
		10010010	10010110	10110110		
	10010000	10010001	10010011	10010111	10110111	
			10111000			
		10011000	10011001	10111001		
		10001010	10011010	10111010		
	10001000	10001001	10001011	10011011	10111011	
		10001100	10011100	10111100		
	10000100	10000101	10001101	10011101	10111101	
	10000010	10000110	10001110	10011110	10111110	
10000000	10000001	10000011	10000111	10001111	10011111	10111111
			11001010			

（续）

			11001000	11001001	11001011			
				11001100				
			11000100	11000101	11001101			
			11000010	11000110	11001110			
		11000000	11000001	11000011	11000111	11001111		
				11010010				
			11010000	11010001	11010011			
				11010100				
			01010100	01010101	11010101			
			01010010	01010110	11010110			
		01010000	01010001	01010011	01010111	11010111		
				11011000				
			01011000	01011001	11011001			
			01001010	01011010	11011010			
		01001000	01001001	01001011	01011011	11011011		
			01001100	01011100	11011100			
		01000100	01000101	01001101	01011101	11011101		
		01000010	01000110	01001110	01011110	11011110		
	01000000	01000001	01000011	01000111	01001111	01011111	11011111	
				11100010				
			11100000	11100001	11100011			
				11100100				
			01100100	01100101	11100101			
			01100010	01100110	11100110			
		01100000	01100001	01100011	01100111	11100111		
				11101000				
			01101000	01101001	11101001			
			00101010	01101010	11101010			
		00101000	00101001	00101011	01101011	11101011		
			00101100	01101100	11101100			
		00100100	00100101	00101101	01101101	11101101		
		00100010	00100110	00101110	01101110	11101110		
	00100000	00100001	00100011	00100111	00101111	01101111	11101111	
				11110000				
			01110000	01110001	11110001			
			00110010	01110010	11110010			
		00110000	00110001	00110011	01110011	11110011		
			00110100	01110100	11110100			
		00010100	00010101	00110101	01110101	11110101		
		00010010	00010110	00110110	01110110	11110110		
	00010000	00010001	00010011	00010111	00110111	01110111	11110111	
			00111000	01111000	11111000			
		00011000	00011001	00111001	01111001	11111001		
		00001010	00011010	00111010	01111010	11111010		
	00001000	00001001	00001011	00011011	00111011	01111011	11111011	
		00001100	00011100	00111100	01111100	11111100		
	00000100	00000101	00001101	00011101	00111101	01111101	11111101	
	00000010	00000110	00001110	00011110	00111110	01111110	11111110	
00000000	00000001	00000011	00000111	00001111	00011111	00111111	01111111	11111111

20

二进位串的集合C叫做杂乱，或者“子集的反链”，如果当σ和τ是C的不同元素时，$\sigma \not\subseteq \tau$，则它的二进位串是不可比较的。埃马努尔 · 斯泊纳[*Math. Zeitschrift* **27**(1928)，544-548]的著名定理断言，在$\{1,\cdots,n\}$上没有能多于M_n个元素的杂乱；而圣诞树模式提供了一个简单的证明，因为无杂乱可以包含每行的一个以上的元素。

的确，圣诞树模式可以用来证明更多的事实。首先让我们注意到，对于$0 \leqslant k \leqslant n/2$，恰有长度$n+1-2k$的$\binom{n}{k}-\binom{n}{k-1}$行存在，因为在列$k$中恰有$\binom{n}{k}$个元素。例如，表4有长度9的一行，即底部那行，它也有长度7的$\binom{8}{1}-\binom{8}{0}=7$行，长度5的$\binom{8}{2}-\binom{8}{1}=20$行，长度3的$\binom{8}{3}-\binom{8}{2}=28$和长度为1的$\binom{8}{4}-\binom{8}{3}=14$行。而且，

这些数 $\binom{n}{k}-\binom{n}{k-1}$ 都出现在卡塔兰三角形(22)中，因为按照等式(23)，它们等于 $C_{k(n-k)}$。

进一步的研究提示，这同卡塔兰的联系不单是一个巧合；事实上，嵌套的括弧是更深入理解圣诞树模式的关键，因为括弧理论告诉我们在哪里一个任意的二进制串可装入数组中。假设分别使用符号(和)来代替1和0，则任何括弧串，无论嵌套与否，对于满足$0\leqslant p\leqslant q$的某个p和q，都可惟一地写成以下形式

$$\alpha_0)\cdots\alpha_{p-1})\,\alpha_p(\alpha_{p+1}\cdots(\alpha_q \tag{37}$$

其中子串α_0，…，α_q适当地嵌套而且可能为空。右括弧的恰好p个和左括弧的恰好$q-p$个，在没有配对的意义下是“自由的”。例如，串

) (( )) ( )) ( )))) (((((( ) (( ) ( ) ((( )) (38)

有p=5，q=12，$\alpha_0=\varepsilon$，α_1=(( )) ( )，α_2=( )，$\alpha_3=\varepsilon$，…，α_{12}=(( ))。一般，串(37)是长度为q+1的链的一部分，

$$\alpha_0)\cdots\alpha_{q-1})\alpha_q,\alpha_0)\cdots\alpha_{q-2})\alpha_{q-1}(\alpha_q,\cdots,\alpha_0(\alpha_1\cdots(\alpha_q \tag{39}$$

其中，以q个自由的)开始，逐一地改成为(。圣诞树模式的每一行恰好以这个方式得到，但使用1和0代替(和)；如果链$\sigma_1\cdots\sigma_s$对应于嵌套的串$\alpha_0\cdots\alpha_{s-1}$，则在(36)中的后继链分别对应于$\alpha_0$，…，$\alpha_{s-3}$，$\alpha_{s-2}(\alpha_{s-1})$和对应于$\alpha_0$，…，$\alpha_{s-3}$，$\alpha_{s-2}$，$\alpha_{s-1}$，$\varepsilon$。[参见库尔梯斯 · 格林和丹尼尔J · 克列特曼，*J. Combinatorial Theory* **A 20**(1976), 80-88]。

进一步注意，在模式每行最右边的元素，例如，在n=8的情况下10101010,10101011,10101100,10101101，…，11111110，11111111是词典顺序。因此，例如表4中长度为1的14行精确地对应于表1中嵌套括弧的14个串。这个发现使得通过类似于算法P的一个方法，从底向上顺序地生成表8的行变得很容易；参
21 见习题77。

令$f(x_1,\cdots,x_n)$是n个变量的单调布尔函数。如果$\sigma=a_1\cdots a_n$是长度为n的任何二进位串，为方便起见，我们可以写$f(\sigma)=f(a_1,\cdots,a_n)$。圣诞树模式的任何行$\sigma_1\cdots\sigma_s$形成一个链，所以我们有

$$0\leqslant f(\sigma_1)\leqslant\cdots\leqslant f(\sigma_s)\leqslant 1 \tag{40}$$

换言之，存在一个下标t，对于$j<t$使$f(\sigma_j)=0$；对于$j\geqslant t$，$f(\sigma_j)=1$；如果对于这个模式的每一行，知道下标t，则将知道对于所有2^n个二进位串σ的$f(\sigma)$的值。

乔治斯 · 汉塞尔[*Comptes Rendus Acad. Sci.* (A) **262**(巴黎，1966)，1088-1090]注意到，圣诞树模式有另一个重要的性质：如果σ_{j-1}，σ_j和σ_{j+1}是任何行的三个连续项，则二进位串

$$\sigma_j'=\sigma_{j-1}\oplus\sigma_j\oplus\sigma_{j+1} \tag{41}$$

位于前一行。事实上，σ_j'位于和σ_j相同的列，而且它满足

$$\sigma_{j-1}\subseteq\sigma_j'\subseteq\sigma_{j+1} \tag{42}$$

在区间$(\sigma_{j-1}\cdots\ \sigma_{j+1})$中叫做$\sigma_j$的相对补。通过归纳法容易证明汉塞尔的发现，因为递归规则(36)定义了圣诞树模式。通过在比较少的仔细选定的位置实际计算函数，对于所有σ，可推导出$f(\sigma)$的值；如果知道$f(\sigma'_j)$的值，根据关系(42)，就将知道$f(\sigma_{j-1})$或$f(\sigma_{j+1})$。

算法H（认识单调布尔函数）。令$f(x_1, \cdots, x_n)$是每个布尔变量非递减的一个布尔函数，但除此之外皆未知。给定长度为n的一个二进位串σ。令$r(\sigma)$是圣诞树模式中出现有σ的行的个数，其中$1 \leqslant r(\sigma) \leqslant M_n$。如果$1 \leqslant m \leqslant M_n$，令$s(m)$是行$m$中二进位串的个数，也令$\chi(m, k)$为对于$(n+1-s(m))/2 \leqslant k \leqslant (n-1+s(m))/2$，该行的列$k$中的二进位串。这个算法确定阈值$t(1), t(2), \cdots, t(M_n)$的序列，使得

$$f(\sigma)=1 \Leftrightarrow \nu(\sigma) \geqslant t(r(\sigma)) \tag{43}$$

在每行中对于f计算不多于两个点。

H1. [对m进行循环。] 对于$m=1, \cdots, M_n$，执行步骤H2～H4，然后停止。

H2. [开始行m。] 置$a \leftarrow (n+1-s(m))/2$和$z \leftarrow (n-1+s(m))/2$。

H3. [进行一个二分查找。] 如果$z \leqslant a+1$，则转到步骤H4。否则，置$k \leftarrow \lfloor (a+z)/2 \rfloor$，而且

$$\sigma \leftarrow \chi(m, k-1) \oplus \chi(m, k) \oplus \chi(m, k+1). \tag{44}$$

如果$k \geqslant t(r(\sigma))$，则置$z \leftarrow k$；否则置$a \leftarrow k$，重复步骤H3。

H4. [计算。] 如果$f(\chi(m, a))=1$，置$t(m) \leftarrow a$；否则，如果$a=z$，置$t(m) \leftarrow a+1$；否则置$t(m) \leftarrow z+1-f(\chi(m, z))$。▌ 22

在最坏情况下也只在最少可能的点计算f的值的意义下，汉塞尔算法是最优的。如果f刚好是阈值函数

$$f(\sigma)=[\nu(\sigma)>n/2], \tag{45}$$

则任何在圣诞树模式的头m行知道f的有效算法必定在每行的列$\lfloor n/2 \rfloor$以及在大小大于1的每行的$\lfloor n/2 \rfloor+1$列计算$f(\sigma)$。否则，不能把f同与它仅在未考察的点处不同的函数区分开来。[参见维·康·柯罗波科夫 *Problemy Kibernetiki* **13**(1965), 5-28，定理5。]

有向树和森林。我们来看另外一类树，其中父子关系是重要的，但是每个家庭中孩子的顺序则不重要。n个节点的一个有向森林可通过一个指针序列$p_1 \cdots p_n$来定义，其中p_j是节点j的父亲（或者，如果j为根，则$p_j=0$）；具有弧$\{j \to p_j \mid 1 \leqslant j \leqslant n\}$的顶点$\{0, 1, \cdots, n\}$的有向图将没有有向环。一棵有向树是恰有一个根的有向森林。(参见2.3.4.2节。)每个n节点的有向森林等价于一棵$(n+1)$个节点的有向树，因为该树的根可以当作森林的所有根的父亲。在2.3.4.4节我们看到，有A_n个带有n个节点的有向树，其中A_n的最初一些值如下：

$n=$	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$A_n=$	1	1	2	4	9	20	48	115	286	719	1842	4766	12486	32973；

(46)

渐近地，$A_n=c\alpha^n n^{-3/2}+O(\alpha^n n^{-5/2})$，其中$\alpha \approx 2.9558$和$c \approx 0.4399$。例如，当我们忽略水

平的自左到右的顺序和仅考虑垂直方向时，在表1的14个森林中仅有9个是不同的。

如果我们使用由胡·伊·斯科因斯所引进的对树的一种排序[*Machine Intelligence* **3**(1968), 43-60]，对每个家庭成员适当地排序，则每一个有向森林对应于惟一排序的森林。从(11)回想起，排了序的森林可通过它们的层次代码$c_1 \cdots c_n$加以表征，其中节点j以前根顺序出现在层c_j上。如果在每个家庭中子树的层次代码序列是非递增的词典次序，则一个排了序的森林称为是规范的。例如，表1中的规范森林是其层次代码$c_1c_2c_3c_4$为0000, 0100 , 0101, 0110, 0111, 0120, 0121, 0122和0123的那些森林。层次序列0112不是规范的，因为这个根的子树有各自的层次代码1和12；串1在词典顺序下小于12。我们通过归纳法可以容易地验证，对于给定的有向森林对子树重新排序的所有方法当中，规范层次代码是在词典顺序下最大的。

特·比伊尔和沙·米·黑迪特尼米[*SICOMP* **9**(1980), 706-712]注意到，如果以规范层次代码的递减词典顺序来访问它们，则将有一个相当简单的方法来生成有向森林。假设$c_1 \cdots c_n$是规范的，其中$c_k>0$和$c_{k+1}=\cdots=c_n=0$。通过减小c_k得到次最小序列，然后把$c_{k+1} \cdots c_n$增加到同规范性相一致的最大层次；而且那些层次是易于
23 计算的。如果$j=p_k$是节点k的父亲，则对于$j<l\leqslant k$，我们有$c_j=c_k-1$。因此，层$c_j \cdots c_k$表示当前以节点j为根的子树。为了得到小于$c_1 \cdots c_n$的最大层序列，以无穷序列$(c_j \cdots c_{k-1})^\infty=c_j \cdots c_{k-1}\, c_j \cdots c_{k-1}\, c_j \cdots$的头$n+1-k$个元素代替$c_k \cdots c_n$。(其效果是从它当前的位置删去$k$作为$j$的最右儿子，然后通过尽可能经常地克隆$j$和它的后裔，来扩展作为$j$的兄弟的新子树。这个克隆过程可能在序列$c_j \cdots c_{k-1}$中间终止，但这不会造成困难，因为一个规范层序列的每个前缀都是规范的。)例如，为了得到以23 443 433 000 000 000结束的规范代码的任何序列的后继，我们以2 344 343 234代替3 000 000 000。

算法O（有向森林）。通过以层次代码$c_1 \cdots c_n$递减的词典顺序访问所有规范n节点森林，这个算法生成n个节点的所有有向森林。然而，不显式地计算这些层次代码；每个规范森林通过在节点的前根顺序下父指针$p_1 \cdots p_n$的序列直接表示。为生成$n+1$个节点的所有有向树，我们想像节点0为根。

O1. [初始化。] 对于$0\leqslant k\leqslant n$，置$p_k\leftarrow k-1$。(特别地，这一步使p_0为非0，用于结束检测；参见步骤O4。)

O2. [访问。] 访问由父指针$p_1 \cdots p_n$表示的森林。

O3. [是容易情况吗？] 如果$p_n>0$，则置$p_n \leftarrow p_{p_n}$，并返回步骤O2。

O4. [求j和k。] 求$p_k\neq 0$的最大的$k<n$。如果$k=0$，则算法终止；否则，置$j\leftarrow p_k$和$d\leftarrow k-j$。

O5. [克隆。] 如果$p_{k-d}=p_j$，则置$p_k\leftarrow p_j$；否则，置$p_k\leftarrow p_{k-d}+d$。如果$k=n$，则返回到步骤O2；否则，置$k\leftarrow k+1$并重复这一步骤。 ▌

如同我们已经看到的其他算法，在步骤O4和O5的循环都十分短；参见习题88。习题90证明，对这个算法稍作改变就足以生成形成自由树的边的所有安排。

生成树。现在来考虑“生成”一个给定图的极小子图。如果G是n个顶点的

一个连通图，G的生成树是不含环的$n-1$条边的子集；同样，它们是形成连接所有顶点的一棵自由树的边的子集。生成树在许多应用中是重要的，特别是在网络的研究中，所以，许多作者都曾处理过生成所有生成树的问题。事实上，在20世纪初期威廉·浮斯纳[*Annalen der Physik* (4), **9**(1902), 1304-1329]在其他人考虑生成其他类型的树之前很久，就已建立了列出所有生成树的系统方法。

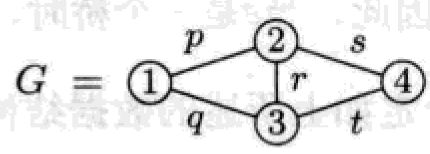
在以下的讨论中，我们将允许图在两个顶点之间有任意数目的边，但不允许从顶点到它自身的循环，因为自循环不能成为树的一部分。浮斯纳的基本思想十分简单，又特别适合于计算：如果e是G的任意边，一棵生成树包含或者不包含e。假设e连接顶点u到顶点v，并假设是生成树的一部分，则该树的其他$n-2$条边生成 24
树把u和v看成相同而得到的图G/e。换句话说，包含e的生成树实质上和把e收缩成一个点时得到的缩小图G/e的生成树相同。另一方面，不含e的生成树是当消去边e时得到的简化图G/e的生成树。因此，从符号上看，G的所有生成树的集合$S(G)$满足

$$S(G)=e\,S(G/e)\cup S(G\backslash e) \tag{47}$$

马尔科姆·詹·史密斯在他的维托里亚大学的硕士论文(1999)中，介绍了以一个“翻转门格雷码”顺序找出所有生成树，来实施递归式(47)的一个漂亮方法。在他的方案中，只需简单地删去一条边而用另一条边代替它，就可从一棵树的前驱得到这棵树。这样的顺序不难找到，但技巧在于要有效地来做这件事。

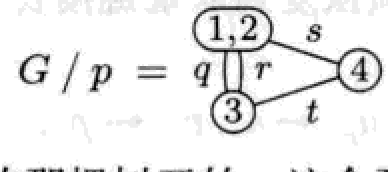
史密斯算法的基本思想是以这样一种方式来生成$S(G)$，即头一棵生成树包括一棵给定的接近树，即不包含环的$n-2$条边的集合。如果$n=2$，则这个任务是平凡的，只需列出所有边即可。如果$n>2$，而且给定的接近树是$\{e_1,\cdots,e_{n-2}\}$，进行如下：假设G是连通的，否则不存在生成树。形成G/e_1，把e_1附加到每棵生成树上，并且以包含$\{e_2,\cdots,e_{n-2}\}$的树开始；注意$\{e_2,\cdots,e_{n-2}\}$是G/e_1的接近树，所以，这个递归有意义。如果以这种方式对于G/e_1找到的最后生成树是$f_1\cdots f_{n-2}$，则通过列出$G\backslash e_1$的所有生成树，以包含接近树$\{f_1,\cdots,f_{n-2}\}$的那一棵树开始，这个任务就完成了。

例如，假设G是下图



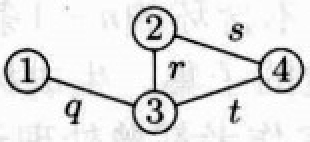
(48)

有4个顶点和5条边$\{p, q, r, s, t\}$。以接近树$\{p, q\}$开始，史密斯过程首先形成收缩图



(49)

列出它的生成树，并以包含q的那棵树开始。这个列表可以是qs，qt，ts，tr，rs；因此，树pqs，pqt，pts，ptr和prs生成G。剩下的任务是列出

$G \setminus p =$  (50)

的生成树，以包含{r, s}的那一棵开始；它们是rsq，rqt，qts。

史密斯算法的详细实现十分有启发性。和通常一样，我们通过令两个弧$u \to v$和$v \to u$对应于每条边u——v来表示图，而且我们维护"弧节点"的表来表示离开每
25 个顶点的弧。我们将需要收缩和解缩图的边，所以，我们将使这些表形成双重链接。如果a指向一个表示$u \to v$的弧节点，则

$a \oplus 1$指向a的"配偶"，它表示$v \to u$；

t_a是a的一个"提示"，即v(因此$t_{a\oplus 1}=u$)；

i_a是任选名，它标识这个边(且等于$i_{a\oplus 1}$)；

n_a指向u的弧表的下一个元素；

p_a指向u的弧表的前一个元素。

l_a不删去弧所用的一个链接，如同以下说明。

顶点以整数{1,⋯,n}表示，而且弧号$v-1$是顶点v的双重链接弧表的一个标题节点。一个标题节点a可由它的提示t_a为0这一事实认记。令d_v是顶点v的度数。因此，例如图(48)可以由$(d_1, d_2, d_3, d_4)=(2, 3, 3, 2)$以及以下弧数据的14个节点表示：

$a =$	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$t_a =$	0	0	0	0	1	2	1	3	2	3	2	4	3	4
$i_a =$					p	p	q	q	r	r	s	s	t	t
$n_a =$	5	4	6	10	9	7	8	0	13	11	12	1	3	2
$p_a =$	7	11	13	12	1	0	2	5	6	4	3	9	10	8

史密斯算法的隐含递归可通过使用弧指针$a_1 \cdots a_{n-1}$的一个数组来方便地控制。在此过程的层次l，弧$a_1 \cdots a_{l-1}$表示已经包括在当前生成树的边；a_l被忽略；而弧$a_{l+1} \cdots a_{n-1}$表示在收缩了的图$(\cdots(G/a_1)\cdots)/a_{l-1}$上的一棵接近树的边，它应是下一棵被访问的生成树的一部分。

还有另一个表示弧栈的弧指针$s_1 \cdots s_{n-2}$的数组，它暂时从当前的图中被删除掉。层次l的栈顶元素是s_l，每个弧a链接到它的后继l_a(在栈的底部它是0)。

如果删除一条边将使一个连通图不连通，那么这条边叫做桥。算法的关键点之一是要保持当前的图连通；因此，当e是一个桥时，我们不设置$G \leftarrow G \setminus e$。

算法S（所有生成树）。给定如上所述的数据结构表示的一个连通图，这个算法访问其所有生成树。

这种称为"跳舞的链接"的技术（将在7.2.2.1节中进行广泛的讨论），在这里用来从双重链接表删除项和把项恢复到双重链接表中。在以下步骤中的省略写法"删除(a)"表示下列的一对操作：

$$n_{p_a} \leftarrow n_a,\ p_{n_a} \leftarrow p_a; \tag{51}$$

类似地，"不删除(a)"表示

26 $$p_{n_a} \leftarrow a,\ n_{p_a} \leftarrow a. \tag{52}$$

S1. [初始化。] 置$a_1 \cdots\ a_{n-1}$到图的一棵生成树上(参见习题94)。并置$x \leftarrow 0$，$l \leftarrow 1$，以及$s_1 \leftarrow 0$。如果n=2，则置$v \leftarrow 1$，$e \leftarrow n_0$并转到S5。

S2. [进入层次l。] 置$e \leftarrow a_{l+1}$，$u \leftarrow t_e$，以及$v \leftarrow t_{e\oplus 1}$，如果$d_u > d_v$，则交换$u \leftrightarrow v$，并置$e \leftarrow e\oplus 1$。

S3. [收缩e。] (现在通过把u的邻接表插进v的表，使u和v相等。我们还必须删除u和v之间所有以前的边，包括e本身在内，否则这样的边将变成环。删去的边链接在一起，以便稍后能在步骤S7中恢复它们。)置$k \leftarrow d_u + d_v$，$f \leftarrow n_{u-1}$，以及$g \leftarrow 0$。在$t_f \neq 0$时，做下列各事：如果$t_f = v$，则删除(f)，删除($f\oplus 1$)，并置$k \leftarrow k-2$，$l_f \leftarrow g$，$g \leftarrow f$；否则，置$t_{f\oplus 1} \leftarrow v$。然后，置$f \leftarrow n_f$，重复这些操作直到$t_f$ =0为止。最后，置$l_e \leftarrow g$，$d_v \leftarrow k$，$g \leftarrow v-1$，$n_{p_f} \leftarrow n_g, p_{n_g} \leftarrow p_f, p_{n_f} \leftarrow g$，$n_g \leftarrow n_f$，以及$a_l \leftarrow e$。

S4. [推进l。] 置$l \leftarrow l+1$。如果$l < n-1$，则置$s_l \leftarrow 0$并返回S2。否则，置$e \leftarrow n_{v-1}$。

S5. [访问。] (当前的图现在仅有两个顶点，其中之一是v。)置$a_{n-1} \leftarrow e$并访问生成树$a_1 \cdots\ a_{n-1}$。(如果x=0，那么这是待访问的头一棵生成树；否则，通过删去x和插入e，它不同于它的前驱。)置$x \leftarrow e$和$e \leftarrow n_e$，如果$t_e \neq 0$则重复这一步。

S6. [减少l。] 置$l \leftarrow l-1$。如果l=0则终止此算法。否则，置$e \leftarrow a_l$，$u \leftarrow t_e$和$v \leftarrow t_{e\oplus 1}$。

S7. [解缩e。] 置$f \leftarrow u-1$，$g \leftarrow v-1$，$n_g \leftarrow n_{p_f}, p_{n_g} \leftarrow g, n_{p_f} \leftarrow f, p_{n_f} \leftarrow f$以及$f \leftarrow p_f$。当$t_f \neq 0$时，置$t_{f\oplus 1} \leftarrow u$和$f \leftarrow p_f$。然后置$f \leftarrow l_e$，$k \leftarrow d_v$，当$f \neq 0$时，置$k \leftarrow k+2$，不删除($f\oplus 1$)，不删除($f$)并置$f \leftarrow l_f$。最后，置$d_v \leftarrow k - d_u$。

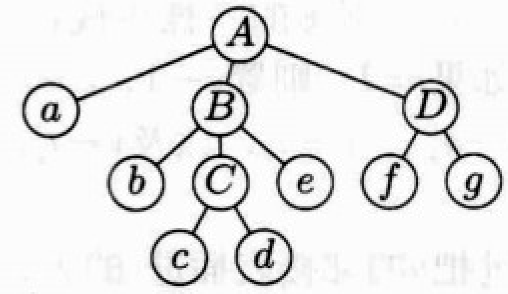
S8. [检测桥。] 如果e是一个桥，则转到S9(关于表示这个检测的一个方法，参见习题95)。否则，置$x \leftarrow e$，$l_e \leftarrow s_l$，$s_l \leftarrow e$，删除(e)和删除($e\oplus 1$)。置$d_u \leftarrow d_u - 1$，$d_v \leftarrow d_v - 1$，并转到S2。

S9. [返工层次l的删除。] 置$e \leftarrow s_l$。当$e \neq 0$时，置$u \leftarrow t_e$，$v \leftarrow t_{e\oplus 1}$，$d_u \leftarrow d_u + 1$，$d_v \leftarrow d_v + 1$，不删除($e\oplus 1$)，不删除($e$)，并且，置$e \leftarrow l_e$。返回到S6。▮

鼓励读者对像(48)这样的小图执行这个算法的所有步骤。注意，如果u的邻接表碰巧变成空的，则在步骤S3和S7中会出现微妙的情况。也要注意可能有若干捷径存在，它以一个更复杂的算法为代价；我们将在本节稍后讨论这样的改进。

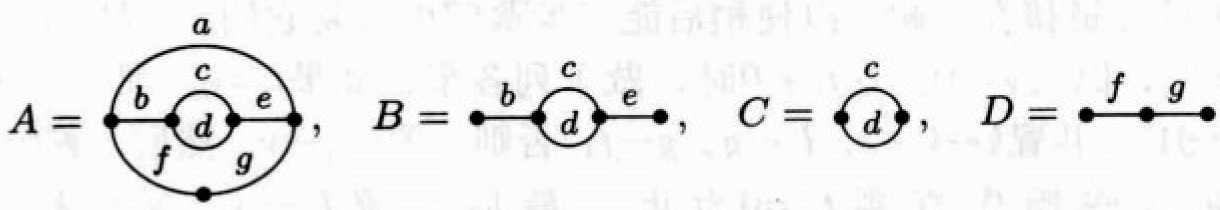
• **串－并行图**。当给定的图有串行和/或并行的分解时，寻找所有生成树的任务将变得特别简单。在s和t之间的一个串－并行图是具有两个指定的顶点s和t的图G，它的边可以递归地构造如下：G由单边s——t组成；或者G是由s_j 和t_j 之间的$k \geqslant 2$个串－并行子图G_j 组成一个*串行超边*，对于$1 \leqslant j < k$ 和$t_k = t$，它们通过$s = s_1$ 和$t_j = s_{j+1}$串 27
行地连接；或者G是在s和t之间并行地连接的$k \geqslant 2$个串－并行子图G_j 组成的一个*并行超边*。给定s和t，如果我们要求对于串行超边的子图G_j 本身不是串行超边，和对于并行超边的子图G_j 本身不是并行超边，则这个分解实质上是惟一的。

任何串－并行图都可以方便地表示为一棵树，其中没有度数为1的节点。这棵树的叶节点表示边，而分支节点表示超边，从一个层次到另一个层次，在串行和并行间交替。例如，如果把顶节点A取作并行树



(53)

对应于串−并行图和子图



(54)

在(54)中对边（而不是顶点）命名，因为相对于生成树边是第一重要的。

假设s和t之间的串−并行图的接近树是不连接s和t的$n-2$个无环的边集合。容易递归地描述生成树和串−并行图的接近树如下：(1) 一棵串行超边的生成树对应于所有它的主要子图G_j的生成树，一棵接近树对应于除一棵G_j之外的所有生成树，一棵接近树在另一个子图中。(2) 一棵并行超边的接近树对应于所有它的主要子图G_j的接近树，一棵生成树对应于除一棵之外所有的G_j的接近树，而一棵生成树在另一个子图中。

规则(1)和规则(2)提议列出串−并行图的生成树和/或接近树的以下数据结构。令p指向像(53)中一棵树中的一个节点。我们定义

$t_p=1$，对串行超边；否则为0(p的"类型")；

$v_p=1$，如果有p的一棵生成树；0，如果有一棵接近树；

$l_p=$ 指向p的最左儿子的指针；或者为0，如果p为一片叶；

$r_p=$ 指向p的右兄弟的指针，循环地包在一起（即两端相连接——译者注）；

$d_p=$ 指向p的一个指定儿子的指针；或为0，如果p为一片叶。

如果q指向p的最右儿子，则它的"右兄弟"r_q等于l_p。如果q指向p的任意儿子，则规则(1)和规则(2)指出

28
$$v_q=\begin{cases}v_p & \text{如果}q=d_p\\ t_p & \text{如果}q\neq d_p\end{cases} \tag{55}$$

(例如，如果p是表示一个串行超边的分支节点，则对于除一个之外所有p的儿子，我们必定有$v_q=1$；惟一的例外是指定的儿子d_p。因此，对于所有串行地连在一起形成p的子图必有一棵生成树，仅对于这种情况中的一个指定的子图是例外，有p的一棵接近树。)

给定指定的儿子指针d_p的任何设置，以及给定对于在树根v_p的任何值0或1，式(55)告诉我们怎样把这些值向下传播到所有的叶上。例如，如果在树(53)中置$v_A\leftarrow 1$，如果指定每个分支节点的最左儿子(使得$d_A=a$，$d_B=b$，$d_C=c$和$d_D=f$)，我们逐次地求得

$$v_a=1,\ v_B=0,\ v_b=0,\ v_C=1,\ v_c=1,\ v_d=0,\ v_e=1,\ v_D=0,\ v_f=0,\ v_g=1 \tag{56}$$

一个叶节点q出现在生成树中当且仅当$v_q=1$；因此，(56)指定(54)中串−并行图A的

生成树$aceg$。

为方便起见，我们说p的配置是它的生成树，如果$v_p=1$的话；而如果$v_p=0$，则是它的接近树。我们希望生成根的所有配置。如果$v_p=t_p$，则称一个分支节点p为"省事"的。即如果一个串行节点的配置是生成树，则它是省事的；如果一个并行节点的配置是接近树，则它是省事的。如果p是省事的，则它的配置是它的儿子们的配置的笛卡儿积，即是儿子们的配置的所有k元组，独立地改变；在省事的情况下，指定的儿子d_p是无关紧要的。但是，如果p不是省事的，则它的配置是取遍所有可能的d_p选择，得到这样的笛卡儿k元组的并。

如果运气好，你可以得到它，但省事节点是相对稀少的。一个非省事节点至多有一个儿子(即指定的儿子)能是省事的。而一个省事节点的所有儿子们都不是省事的，除非它们是叶。

尽管如此，一棵串-并行图的树表示使得所有它的生成树和/或接近树的递归生成十分直截和有效。算法S的操作——收缩和解缩、删除和不删除、桥的探测——当处理串-并行图时都不必要。而且，习题99说明，如同我们前面已见过的若干算法那样，通过使用焦点指针，有一个令人高兴的方法可以得到在一棵翻转门格雷码顺序下的生成树或接近树。

*** 算法S的求精。** 尽管算法S为我们提供了访问一般图的所有生成树的一个简单和相当有效的方法，但它的提出者马尔科姆·史密斯认识到，串-并行图的性质可用来使它变得更好些。例如，如果一个图存在相同的顶点u和v之间连接的两个或更多的边，我们可以把它们组合成为一个超边；于是原来的图的生成树可以容易地从那些更简单的归约了的图得到。而且，如果一个图有度数为2的一个顶点v，使得接触v的惟一边是u——v和u——w，则我们可以消除v和通过u和w之间的一条超边来代替这些边。而且，度数为1的任何顶点，连同它的邻边一起，通过简单地把该边包括于每棵生成树中，都可有效地消除掉。

在把上一段中所述的归约应用到一个给定的图G之后，我们得到没有并行边和没有度数为1或2的顶点的一个归约了的图$\hat{G}$，而且和$m\geqslant0$个串-并行图$S_1,\cdots,S_m$的一个集合一起，表示必须包括在G的所有生成树中的边(或超边)。事实上，$\hat{G}$的每个剩下的边u——v对应于顶点u和v之间的一个串-并行图S_{uv}。于是，G的生成树是作为取遍$\hat{G}$的所有生成树T，取遍S_1，…，S_m的生成树的笛卡儿积的生成树，以及对于T中的边u——v的所有S_{uv}的生成树的并得到的，连同在$\hat{G}$中但不在T中的边u——v的所有S_{uv}的接近树一起的并而得到。通过使用算法S的策略可以得到$\hat{G}$的所有生成树T。 29

事实上，当以这种方式推广算法S时，以G/e或$G\backslash e$代替当前图G的操作典型地进一步引发归约，因为新的并行边出现或者因为一个顶点的度数降到3之下。因此，结果是，算法S当中隐含的递归的"停止状态"，即只剩下两个顶点的情况(步骤S5)，决不真正出现：一个归约的图$\hat{G}$只有一个顶点且没有边，或者它至少有4个顶点和6条边。

得到的算法保留了期望的算法S的翻转门性质，而且它十分好（尽管它的运行时间大约是原来的4倍）；参见习题100。史密斯证明，它有最好可能的渐近运行时间：如果G有n个顶点，m条边，和N棵生成树，则算法在$O(m+n+N)$步内全部访问完它们。

算法S和加强了效率的算法S'的性能，可通过生成典型图的生成树时实际访问的内存次数，进行最好的评价，如表5所示。该表的最下面一行对应于斯坦福图库的平面−哩数(16,0,0,1,0,0,0)图，它作为前面几行的纯粹数学例子的“有机”矫正方法。在倒数第二行上的随机多图也来自斯坦福图库，通过它的正式名字随机−图(16,37,1,0,0,0,0,0,0,0)更精确地加以描述。尽管4×4圆环体同构于4立方体(参见习题7.2.1.1-17)，但这些同构图产生稍微不同的运行时间，因为当算法运行时，它们遇到的顶点和边不同。

一般，我们可以说，算法S对于小的例子不太差，仅当这个图十分稀疏时才例外；但是，当有许多生成树出现时，算法S'开始崭露头脚。一旦算法S'热了身，它就趋向于在进行18次或19次内存访问之后产生一棵新树。

表5也指出，一个数学上定义的图通常有一棵令人惊奇的生成树的“约略”数。例如，德·姆·克维特奇[*Srpska Akademija Nauka Matematicheski Institut* **11**(贝尔格莱德：1971)，135-141]在其他结果中还发现，n立方体恰有

$$2^{2^n-n-1}1^{\binom{n}{1}}2^{\binom{n}{2}}\cdots n^{\binom{n}{n}} \tag{57}$$

30 个。习题104~149剖析为什么会出现这个情况的一些原因。

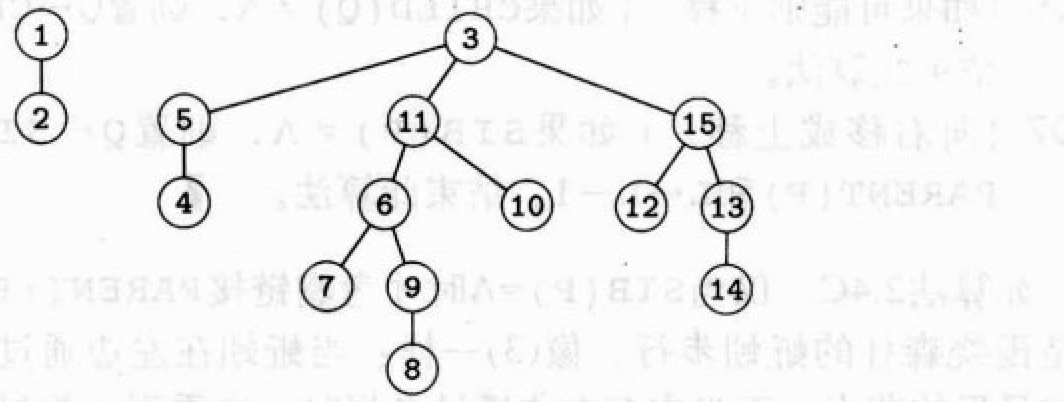
表5 为生成所有生成树所需的内存访问的运行时间

	m	n	N	算法S	算法S'	每个树的μ	
路径P_{10}	9	10	1	794μ	473μ	794.0	473.0
路径P_{100}	99	100	1	9 974μ	5 063μ	9974.0	5063.0
环C_{10}	10	10	10	3 480μ	998μ	348.0	99.8
环C_{100}	100	100	100	355 605μ	10 538μ	3556.1	105.4
完全图K_4	6	4	16	1 213μ	1 336μ	75.8	83.5
完全图K_{10}	45	10	100 000 000	3 759.58Mμ	1 860.95Mμ	37.6	18.6
完全二分图$K_{5,5}$	25	10	390 625	23.43Mμ	8.88Mμ	60.0	22.7
4×4栅格$P_4\times P_4$	24	16	100 352	12.01Mμ	1.87Mμ	119.7	18.7
5×5栅格$P_5\times P_5$	40	25	557 568 000	54.68Gμ	10.20Gμ	98.1	18.3
4×4圆柱体$P_4\times C_4$	28	16	2 558 976	230.96Mμ	49.09Kμ	90.3	19.2
5×5圆柱体$P_5\times C_5$	45	25	38 720 000 000	3 165.31Gμ	711.69Gμ	81.7	18.4
4×4圆环体$C_4\times C_4$	32	16	42 467 328	3 168.15Mμ	823.08Mμ	74.6	19.4
4立方体$P_2\times P_2\times P_2\times P_2$	32	16	42 467 328	3 168.16Mμ	823.38Mμ	74.7	19.4
随机多图	37	16	59 933 756	3 818.19Mμ	995.91Mμ	63.7	16.6
16城市	37	16	179 678 881	11 772.11Mμ	3 267.43Mμ	65.5	18.2

一般的亚−格雷码。让我们通过讨论完全不同的，但还是与树有关的问题来结束本节。考虑以下遍历一个非空森林的两个标准方法的混合变形。

前后根顺序的遍历	后前根顺序的遍历
访问第一棵树的根	以前后根顺序遍历第一棵树的子树
以后前根顺序遍历第一棵树的子树	访问第一棵树的根
以前后根顺序遍历剩下的树	以后前根顺序遍历剩下的树

在第一种情况下，以前后根顺序遍历森林的每一棵树，首先访问它的根；但是以后前根顺序遍历这些根的子树，最后访问根。第二个方案类似，但把“前”和“后”对换。而且一般说来，对于森林的每个偶编号的层，前后根顺序首先访问根，但对于奇编号的层，则最后访问它们。例如，(2)中的森林在我们以前后根顺序对它的节点进行标号时，变成



(58)

31

前后根顺序和后前根顺序不仅有奇特性，而且实际上很有用。原因在于，以任意顺序相邻节点在森林中总是彼此接近的。例如，对于k=1, 4, 6, 8, 10, 13，节点k和k+1在(58)中是相邻的；当k=3,12,14时，它们仅被一个节点分开；而当k=2, 5, 7, 9, 11时，它们相隔三步(如果我们想像在森林顶部有一个看不见的超父亲的话)。稍加思索就可递归地证明，至多有两个节点可能在前后根顺序的邻居或后前根顺序的邻居之间产生干扰——因为后前根顺序(F)总是从第一棵树的根或者它的最左儿子开始，而前后根顺序(F)总是以最后一棵树的根或它的最右儿子结束。

假设我们要生成某种类型的所有组合模式，而且要以类格雷码的方式访问它们，使得连续的模式总是彼此“接近”。则至少在概念上，我们能够形成所有可能模式p的图，连通对于彼此相互接近的所有模式都有边p——q。以下由米兰 · 斯卡尼纳[*Spisy Přírodovědecké Fakulty University v Brně* No. 412(1960), 137-142]给出的定理证明，只要通过一序列的短步骤，我们可以从任何模式得到其他模式，则一个相当好的格雷码总是可能的。

定理S。任何连通图的顶点可以以一个循环顺序$(v_0, v_1, \cdots, v_{n-1})$列出，这样，对于$0 \leqslant k<n$，$v_k$ 和$v_{(k+1) \bmod n}$之间的距离至多为3。

证明。寻找图中的一棵生成树，并且以前后根顺序遍历它。 ▮

传统上，图论专家说，一个图G的第k次方是图G^k，其顶点是G中的那样一些顶点，使u——v在G^k中当且仅当在G中从u到v有长度为k或更短的一条通路。因此当n>2时，他们能够更简明地表达定理S：一个连通图的立方体是哈密顿图。

当我们以无循环的方式访问一棵树的节点时，前后根顺序的遍历也是有用的，而且在各次停止之间的步骤数是有界的。

算法Q（在三重链接的森林里前后根顺序的后继）。如果P指向由链接PARENT、CHILD和SIB所表示的一个森林中的一个节点，它们对应于每个节点的父亲、最左儿子和右兄弟，则这个算法计算在前后根顺序下P的后继节点Q。我们假定知道P在森林中出现的层L；L的值更新成为Q的层。如果P刚好是前后根顺序下的最后节点，则此算法置Q←Λ和L←−1。

Q1. [前还是后？] 如果L是偶数，则转到Q4。

Q2. [继续后前根顺序。] 置Q←SIB(P)。如果Q≠Λ，则转到Q6。

Q3. [向上移。] 置P←PARENT(P)，并且L←L−1。转到Q7。

Q4. [继续前后根顺序。] 如果CHILD(P)=Λ，则转到Q7。

Q5. [下移。] 置Q←CHILD(P)和L←L+1。

Q6. [如果可能则下移。] 如果CHILD(Q)≠Λ，则置Q←CHILD(Q)和L←L+1。
32 结束此算法。

Q7. [向右移或上移。] 如果SIB(P)≠Λ，则置Q←SIB(P)；否则置Q←PARENT(P)和L←L−1。结束此算法。 ▮

注意，如算法2.4C，仅当SIB(P)=Λ时才考察链接PARENT(P)。一个完全遍历实际上是围绕森林的蚯蚓步行，像(3)一样：当蚯蚓在左边通过各层时，它“看到”偶数编号层的节点，而当它在右边通过各层时，它看到奇数层的节点。

习 题

1. [*15*] 如果一条蚯蚓围绕二叉树(4)爬行，它如何能容易地重新构造(1)的括弧？

2. [*20*] (斯·扎斯(S. Zaks)，1980)修改算法P使得它产生(8)的组合$z_1 z_2 \cdots z_n$，以代替括弧串$a_1 a_2 \cdots a_{2n}$。

▶3. [*23*] 证明(11)把$z_1 z_2 \cdots z_n$转换成为反演表$c_1 c_2 \cdots c_n$。

4. [*20*] 真或假？如果在词典顺序下产生串$a_1 \cdots a_{2n}$，则对应的串$d_1 \cdots d_n$，$z_1 \cdots z_n$，$p_1 \cdots p_n$和$c_1 \cdots c_n$也如此。

5. [*15*] 什么表$d_1 \cdots d_n$，$z_1 \cdots z_n$，$p_1 \cdots p_n$和$c_1 \cdots c_n$对应于嵌套的括弧串(1)？

▶6. [*20*] 什么匹配对应于(1)？(参看表1的最后列。)

7. [*16*] (a)当算法P结束时，串$a_1 a_2 \cdots a_{2n}$的状态是什么？(b)当算法B结束时，数组$l_1 \cdots l_n$和$r_1 \cdots r_n$包含什么？

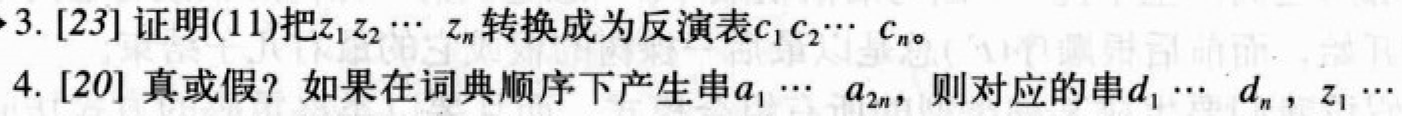
8. [*15*] 什么表$l_1 \cdots l_n$，$r_1 \cdots r_n$，$e_1 \cdots e_n$和$s_1 \cdots s_n$对应于例子森林(2)？

9. [*M20*] 证明表$c_1 \cdots c_n$和$s_1 \cdots s_n$通过以下定律相关联

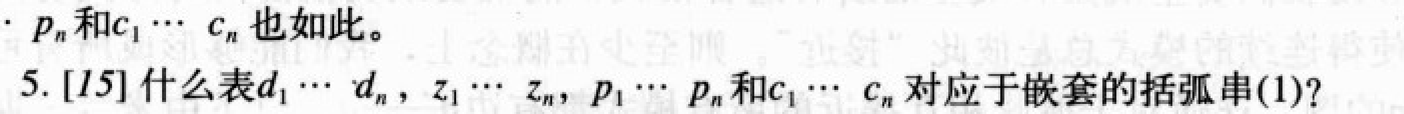
$$c_k = [s_1 \geqslant k-1]+[s_2 \geqslant k-2]+\cdots+[s_{k-1} \geqslant 1]$$

10. [*M20*] (蚯蚓步行。)给定嵌套括弧串$a_1 a_2 \cdots a_{2n}$，对于$0 \leqslant j \leqslant 2n$，令$w_j$是在$a_1 a_2 \cdots a_j$中左括弧超出右括弧的个数。证明$w_0+w_1+\cdots+w_{2n}=2(c_1+\cdots+c_n)+n$。

11. [*11*] 如果F是一个森林，则它的*共轭*F^R通过自左至右的镜面反射得到。例如，表1中的14个森林是



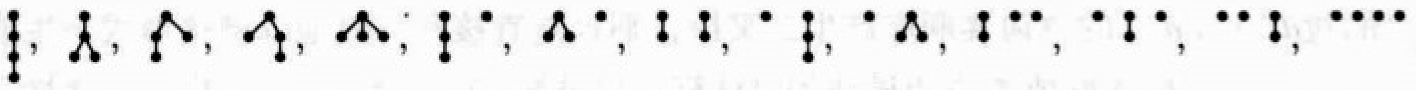
它们的共轭如表2的协词典顺序的森林，分别是



如果F对应于嵌套的括弧$a_1 a_2 \cdots a_{2n}$，什么括弧串对应于F^R？

12. [*15*] 如果F是一个森林，它的*转置*F^T是这样一个森林，它的二叉树通过表示F的二

叉树中交换左和右链接而得到。例如，表1中的14个森林的转置分别为



森林(2)的转置是什么？

13. [*20*] 继续习题11和12，一个标了号的森林F的前根顺序和后根顺序如何与(a)F^R，(b)F^T的前根顺序和后根顺序相关联。 33

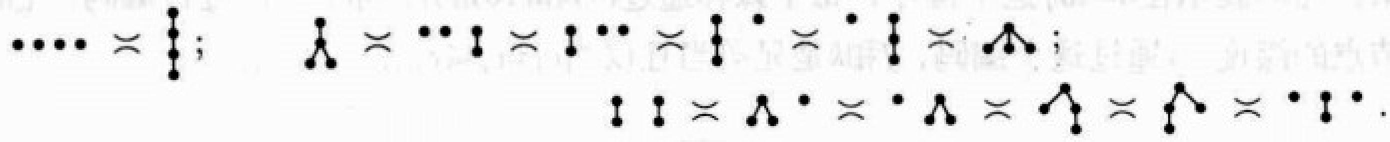
▶14. [*21*] 求出所有加了标号的森林F，使得$F^{RT}=F^{TR}$。

15. [*20*] 假设B是如习题2. 3. 2-5和表2最后一行，把每个节点同它的左兄弟和最右儿子链接，从森林F得到的一棵二叉树。设F'是经由左儿子和右兄弟的链接，以通常的方式对应于B的森林。试证明，在习题11和12的记号下，$F'=F^{RT}$。

16. [*20*] 如果F和G是森林，令FG是通过把F的树放置在G的树的左边而得到的森林，并且令$F \mid G=(G^T F^T)^T$。试给出操作符$\mid$的一个直观说明，并证明它是结合的。

17. [*M46*] 表征所有未加标号的森林F，使得$F^{RT}=F^{TR}$。(参见习题14)。

18. [*30*] 说两个森林是同族的，如果通过重复的取共轭和/或转置的操作，可以从一个森林获得另一个森林。习题11和12的例子表明，4个节点的所有森林属于3个同族之一：



试研究具有15个节点的所有森林。它们一共形成多少个同族森林的等价类？最大的类是什么？最小的类是什么？包含(2)的类的大小是什么？

19. [*28*] 令$F_1, F_2, \cdots, F_N$为对应于由算法P生成的嵌套括弧的未加标号的森林序列，并令$G_1, G_2, \cdots, G_N$是对应于由算法B生成的二叉树的未加标号的森林序列。试证明，在习题11和12的记号下，$G_k=F_k^{RTR}$。(森林F^{RTR}称为森林F的对偶；在以下的若干习题中，它记为F^D)。

20. [*25*] 回忆2. 3节，在一棵树中一个节点的度数是它具有的儿子的个数，一棵扩展的二叉树以每个节点的度数为0或2这一性质为特征。在扩展的二叉树(4)中，在前根顺序下节点的度数序列为22002220022202200020022022000000，这个0和2的串和(1)中括弧的序列是相同的，除了每个(用2代替，而每个)用0代替之外，另外还附加了一个0。

a) 试证明非负整数$b_1b_2\cdots b_N$的一个序列是一个森林的前根顺序度的序列当且仅当对于$1 \leqslant k \leqslant N$，它满足以下性质：

$$b_1+b_2+\cdots+b_k+f>k \qquad \text{当且仅当} k<N.$$

这里$f=N-b_1-b_2-\cdots-b_N$是森林中的树的棵数。

b) 由习题2. 3. 4. 5－6可回想起，一棵扩展的三叉树以下列性质为特征，每一节点有0或3的度；n个内节点的一棵扩展的三叉树有$2n+1$个外部节点。因此，总共有$N=3n+1$个节点。通过以词典顺序生成相关联序列$b_1 b_2 \cdots b_N$，试设计一个算法生成具有n个内部节点的所有三叉树。

▶21. [*26*] (斯 · 扎斯和丹 · 理查德，1979)继续习题20，说明如何生成有$N=n_0+\cdots+n_t$个节点，且恰有n_j个度数为j的节点的所有森林的前根顺序的度数序列。例如，当$n_0=4$，$n_1=n_2=n_3=1$和$t=3$时，有效序列$b_1 b_2 b_3 b_4 b_5 b_6 b_7$为

1203000, 1230000, 1300200, 1302000, 1320000, 2013000, 2030010, 2030100, 2031000, 2103000, 2130000, 2300010, 2300100, 2301000, 2310000, 3001200, 3002010, 3002100, 3010200, 3012000, 3020010, 3020100, 3021000, 3100200, 3102000, 3120000, 3200010, 3200100, 3201000, 3210000. 34

▶22. [*30*] (詹 · 科尔斯 (J. Korsh)，2004)作为算法B的一个替换，试证明，如果以在(9)中定义的数$d_1, \cdots, d_{n-1}$的协词典顺序产生二叉树，则也能直接和有效地以链接形式来生成二叉树。($d_1, \cdots, d_{n-1}$的实际值不应明显地加以计算；但链接$l_1 \cdots l_n$ 和$r_1 \cdots r_n$ 应该以这样一种方式加以操作，即我们得到逐次地对应于$d_1 d_2 \cdots d_{n-1}$=000…0, 100…0, 010…0, 110…0, 020…0, 001…0…, 000…$(n-1)$的二叉树。)

▶23. [*25*] (a)由算法N访问的最后一个串是什么？(b)由算法L访问的最后二叉树或森林是什么？提示：参考以下习题40。

24. [*22*] 使用表3的记号，什么序列$l_0 l_1 \cdots l_{15}$，$r_1 \cdots r_{15}$，$k_1 \cdots k_{15}$，$q_1 \cdots q_{15}$和$u_1 \cdots u_{15}$对应于二叉树(4)和森林(2)?

▶25. [*30*] (剪枝和嫁接。)如算法B那样表示二叉树，试设计以这样一种方式访问所有链接表$l_1 \cdots l_n$和$r_1 \cdots r_n$的一个算法，即在访问之间，对于某个下标j，恰好有一个链接从j变为0，而另一个从0变为j。(换言之，每一步从二叉树中删去某个子树j，把它放在别处，并保持前根顺序。)

26. [*M31*] (克里维拉斯格)令F 和F' 是节点以前根顺序编号为1到n的n个节点的森林。我们写$F \ltimes F'$ (“F 结合F'”)，对于$1 \le j < k \le n$，如果每当在F' 中j和k是兄弟时，则它们在F 中也是兄弟。图39表示在n=4时这个偏序；每个森林通过(9)和(10)的序列$c_1 \cdots c_n$进行编码，它确定每个节点的深度。(通过这个编码，j和k是兄弟当且仅当$c_j = c_k \le c_{j+1}, \cdots, c_{k-1}$。)

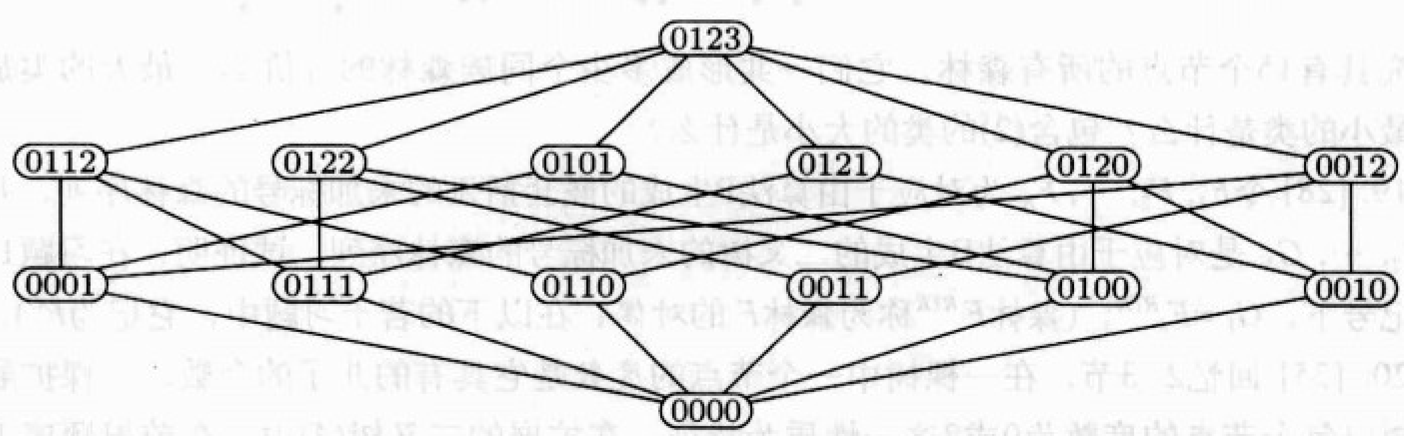


图39 4阶Kreweras格。每个森林由在前根顺序下节点深度的序列$c_1 c_2 c_3 c_4$来表示。(参见习题26~28)

a) 令Π是{1,…,n}的一个分划。试证明存在前根顺序下标号为(1,…,n)的节点的森林F，而且

$$j \equiv k \pmod{\Pi} \Leftrightarrow j\text{是}F\text{ 中}k\text{的一个兄弟}$$

当且仅当Π满足非交叉的性质，即

$$i<j<k<l\text{和}i \equiv k\text{及}j \equiv l \pmod{\Pi}\text{意味着}i \equiv j \equiv k \equiv l \pmod{\Pi}.$$

b) 给定两个n节点的任何森林F 和F'。试说明如何计算它们的最小上界$F \veebar F'$ ，使$F \ltimes G$和$F' \ltimes G$当且仅当$F \veebar F' \ltimes G$的它们的元素。

c) 相对于关系$\ltimes$，何时F' 覆盖F？(参见习题7. 2. 1. 4-55)。

d) 试证明如果F' 覆盖F ，则它恰好比F 少一片叶。

e) 对于$1 \le k \le n$，节点k有e_k 个儿子时，有多少森林覆盖F?

35 f) 使用习题19的对偶性定义，森林(2)的对偶是什么？

g) 证明$F \ltimes F'$ 成立当且仅当$F'^D \ltimes F^D$。(由于这个性质，对偶元素已经关于图39的中心对称地放置)。

h) 给定两个n节点的任何森林F 和F'，试说明如何计算它们的最大下界$F \barwedge F'$? 即$G \ltimes F$和$G \ltimes F'$当且仅当$G \ltimes F \barwedge F'$。

i) 这个格满足类似于习题7. 2. 1. 5-12(f)的半模态法则吗？

▶27. [*M33*] (塔马利格)。继续习题26，我们写$F \dashv F'$，如果对于所有j，在前根顺序下的第j个节点在F'中和在F 中有同样多的后裔。换言之，如果F 和F'通过如表2中的范围序列$s_1 \cdots s_n$和$s'_1 \cdots s'_n$表示，则我们有$F \dashv F'$当且仅当对于$1 \leqslant j \leqslant n$，$s_j \leqslant s_j'$。(参见图40)。

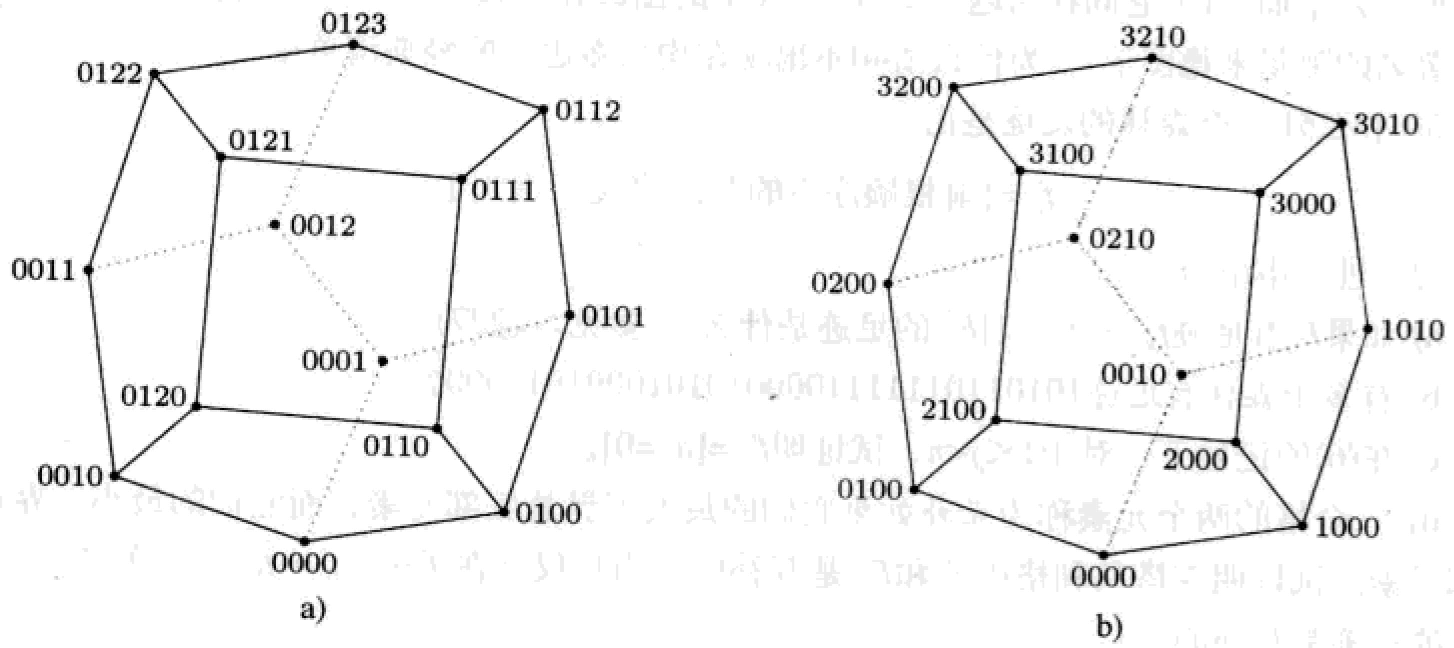


图40 4阶Tamari格。每个森林以前根顺序 a) 节点深度序列及b)后裔计数序列表示。(参见习题26~28)

a) 试证明，范围坐标min (s_1, s'_1) min $(s_2, s'_2) \cdots$min (s_n, s'_n)定义作为F 和F' 的最大下界的森林。(我们以$F \perp F'$ 记之)。提示：证明$s_1 \cdots s_n$ 对应于一个森林当且仅当$0 \leqslant k \leqslant s_j$ 意味着对于$0 \leqslant j \leqslant n$，如果我们定义$s_0 = n$，则$s_{j+k} + k \leqslant s_j$。

b) 在这个偏序下，什么时候F' 覆盖F?

c) 试证明 $F \dashv F'$ 当且仅当 $F'^D \dashv F^D$. 。(同习题26(g)作比较。)

d) 给定F和F'，说明如何计算一个最低上界$F \top F'$。

e) 试证明在克里维拉斯格中的$F \ltimes F'$ 意味着在塔马利格中$F \dashv F'$。

f) 真或假：$F \barwedge F' \dashv F \perp F'$。

g) 真或假：$F \veebar F' \ltimes F \top F'$。

h) 当通路的每个森林覆盖它的后继时，从塔马利格的顶部到底部最长和最短的通路是什么？(这样的通路在格中叫做极大链，试同习题7.2.1.4-55(h)作比较)。

28. [*M26*] (斯坦利格。)继续习题26和27，我们再定义n节点森林的另一个偏序，每当对于$1 \leqslant j \leqslant n$，深度坐标$c_1 \cdots c_n$ 和 $c'_1 \cdots c'_n$ 满足$c_j \leqslant c'_j$时，$F \subseteq F'$。(参见图41。)

a) 通过说明如何计算两个给定森林的最大下界$F \cap F'$ 和最小上界$F \cup F'$，证明这个偏序是一个格。

b) 证明斯坦利格满足分配律

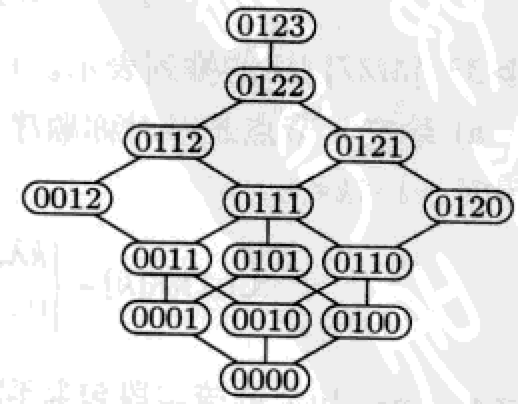


图41 4阶斯坦利格。每个森林以前根顺序节点深度序列表示。(参见习题26~28)

36 $$F\cap(G\cup H)=(F\cap G)\cup(F\cap H),\qquad F\cup(G\cap H)=(F\cup G)\cap(F\cup H)$$

c) 在这个格中何时F'覆盖F？

d) 真或假：$F\subseteq G$ 当且仅当 $F^R\subseteq G^R$。

e) 试证明每当在塔马利格中 $F\dashv F'$ 时，则在斯坦利格中$F\subseteq F'$。

29. [*HM31*] 塔马利格的覆盖图有时叫做一个"结合几何体"，因为它同结合律(14)有关，这在习题27(b)中证明过了。图40中所描述的4阶结合几何体，看上去好像它有3个平面和正五边形的六个面。(把它同在习题7. 2. 1. 2-60中的图23作比较，它显示4阶的"排列晶面"，一个著名的阿基米德图形)。为什么图40不出现在均匀多边形的经典清单中？

30. [*M26*] 一个森林的足迹是由

$$f_j=[\text{前根顺序下的节点}j\text{不是一片叶}]$$

定义的二进位串$f_1\cdots f_n$。

a) 如果F有足迹$f_1\cdots f_n$，则F^D的足迹是什么？(参见习题27)

b) 有多少森林有足迹1010110111110000101010001011000？

c) 在(6)的记号下，对于$1\leqslant j<n$，试证明$f_j=[d_j=0]$。

d) 一个格的两个元素称为互补如果它们的最大下界是底部元素，而它们的最小上界是顶部元素。试证明在塔马利格中F和F'是互补的，当且仅当在 $f_1'\cdots f_{n-1}'=\bar{f}_1\cdots\bar{f}_{n-1}$ 的意义下，它们的足迹是互补的。

▶31. [*M28*] 有n个内部节点的二叉树是退化的，如果它有$n-1$的高度。

a) 有多少n节点二叉树是退化的？

b) 在表1、2和3中我们已经看到，二叉树和森林可以各种n元组的数来编码。对于每个编码$c_1\cdots c_n$，$d_1\cdots d_n$，$e_1\cdots e_n$，$k_1\cdots k_n$，$p_1\cdots p_n$，$s_1\cdots s_n$，$u_1\cdots u_n$和$z_1\cdots z_n$，试说明怎样一看就知道相应的二叉树是退化的。

c) 真或假：如果F是退化的，则F^D也是。

d) 试证明，如果F和F'是退化的，则 $F\curlywedge F'=F\perp F'$ 以及 $F\curlyvee F'=F\top F'$. 也是。

▶32. [*M30*] 试证明，如果 $F\dashv F'$，则存在一个森林 F'' 对于所有G，我们有

$$F'\perp G=F \text{ 当且仅当 } F\dashv G\dashv F''.$$

结果，在塔马利格中半分配律成立：

$$F\perp G=F\perp H \text{ 意味着 } F\perp(G\top H)=F\perp G$$

37 $$F\top G=F\top H \text{ 意味着 } F\top(G\perp H)=F\top G$$

▶33. [*M27*] (树的排列表示。) 令σ是循环$(12\cdots n)$。

a) 给定其节点是以对称顺序编号为1至n的二叉树，试证明，存在$\{1\cdots n\}$的惟一排列λ，使得对于$1\leqslant k\leqslant n$，

$$\mathtt{LLINK}[k]=\begin{cases}k\lambda, & \text{如果}k\lambda<k\\ 0, & \text{否则}\end{cases}\qquad \mathtt{RLINK}[k]=\begin{cases}k\sigma\lambda, & \text{如果}k\sigma\lambda>k\\ 0, & \text{否则}\end{cases}$$

因此λ几乎把$2n$个链接字段包装到一个n元数组中。

b) 试证明当二叉树是森林F的左兄弟/右儿子表示时，这个排列λ特别容易以循环形式来描述。当F是(2)中的森林时，什么是$\lambda(F)$的循环形式？

c) 试找出$\lambda(F)$和对偶排列$\lambda(F^D)$之间的简单关系。

d) 试证明，在习题26中，F' 覆盖F 当且仅当$\lambda(F')=(j\ k)\lambda(F)$，其中$j$和$k$是$F$ 中的兄弟。

e) 结果，在n阶克里维拉斯格中极大链的个数为把n循环分解为$n-1$个转置的乘积的方法数。试计算这个数。提示：参看等式1. 2. 6-(16)。

34. [*M25*] (R. P. 斯坦利)试证明在n阶斯坦利格中极大链的个数为

$$(n(n-1)/2)!/(1^{n-1}3^{n-2}\cdots(2n-5)^2(2n-3)^1)$$

35. [*HM37*] (道 · 布 · 梯勒和迪 · 罗 · 希克森。)试说明为什么渐近公式(16)的分母都是2的幂？

▶ 36. [*M25*] 试分析习题20(b)的三叉树生成算法。提示：由习题2.3.4.4-11，共有 $(2n+1)^{-1}\binom{3n}{n}$ 个具有n个内部节点的三叉树。

▶37. [*M40*] 试分析用于生成给定度数分布$n_0,n_1,n_2,\cdots,n_t$ 的所有树的扎斯-理查德算法(习题21)。提示：参见习题2. 3. 4. 4-32。

38. [*M22*] 作为n的一个函数，由算法L所执行的内存访问的总次数是什么？

39. [22] 通过证明(5)中A_{pq} 的元素对应于带有2行的杨氏表景，来证明公式(23)。

40. [*M22*] (a)在p和$q+1$的二进制表示中没有公共的二进位这一意义下，证明C_{pq} 为奇数当且仅当$p\&(q+1)=0$。(b)因此，C_n为奇数当且仅当$n+1$是2的幂。

41. [*M21*] 试证明选票数有一个简单的生成函数 $\sum C_{pq}w^pz^q$ 。

▶42. [*M22*] 有n个节点的未标号森林中有多少是(a)自共轭的？(b)自转置的？(c)自对偶的？(参见习题11，12，19和26。)

43. [*M21*] 试借助于卡塔兰数〈$C_0,C_1,C_2,\cdots$〉来表达C_{pq}，目的在于当$q-p$很小时给出一个简单公式。(例如：$C_{(q-2)q}=C_q-C_{q-1}$。)

▶44. [*M27*] 试证明，算法B对于每棵访问的二叉树只做 $8\frac{2}{3}+O(n^{-1})$次对内存的访问。

45. [*M26*] 试分析由习题22中的算法所做的内存访问。同算法B相比，它怎么样？

46. [*M30*] (广义的卡塔兰数)。通过如下定义推广(21)：

$$C_{pq}(x)=C_{p(q-1)}(x)+x^{q-p}C_{(p-1)q}(x),\quad 如果0\leqslant p\leqslant q\neq 0；C_{00}(x)=1$$

以及$C_{pq}(x)=0$如果$p<0$或$p>q$；于是$C_{pq}=C_{pq}(1)$。还令$C_n(x)=C_{nn}(x)$，使得〈$C_0(x), C_1(x),\cdots$〉= 〈$1,1,1+x,1+2x+x^2+x^3,1+3x+3x^2+3x^3+2x^4+x^5+x^6,\cdots$〉。 38

a) 试证明$[x^k]C_{pq}(x)$是在(28)中从ⓟⓠ到⓪⓪的通路数，这些通路有区域k，其中一条通路的"区域"是在它上边的矩形单元的个数(因此一条L形状的通路有极大可能的区域 $p(q-p)+\binom{p}{2}$ 。)

b) 试证明 $C_n(x)=\sum_F x^{c_1+\cdots+c_n}=\sum_F x^{内部通路长度(F)}$，对所有有$n$节点森林$F$ 求和。

c) 如果 $C(x,z)=\sum_{n=0}^{\infty}C_n(x)z^n$ 。试证明$C(x,z)=1+zC(x,z)C(x,xz)$。

d) 此外，$C(x,z)C(x,xz)\cdots C(x,x^rz)=\sum_{p=0}^{\infty}C_{p(p+r)}(x)z^p$ 。

47. [*M27*] 继续上一题，推广等式(27)。

48. [*M25*] (弗 · 拉斯基和安 · 普罗斯库洛夫斯基)当$x=-1$时计算$C_{pq}(x)$，并使用这个结

果证明当$n\geqslant5$为奇数时，不可能有嵌套括弧的“完美”格雷码存在。

49. [*17*] 15个嵌套括弧对的词典顺序下的第100万个串是什么？

50. [*20*] 试设计算法U的逆：给定嵌套括弧串$a_1\cdots a_{2n}$，试确定在词典顺序下它的秩$N-1$。什么是(1)的秩？

51. [*M22*] 设$\bar{z}_1\bar{z}_2\cdots\bar{z}_n$是相对于$2n$的$z_1\cdots z_n$的补；换言之，$\bar{z}_j=2n-z_j$，其中$z_j$在(8)中定义。试证明，如果$\bar{z}_1\bar{z}_2\cdots\bar{z}_n$是由算法7. 2. 1. 3L生成的$\{0,1,\cdots,2n-1\}$的第$N+1$个$n$组合，则$z_1 z_2\cdots z_{2n}$是由习题2的算法所生成的$\{1, 2, \cdots, 2n\}$的第$(N-\kappa_n N+1)$个$n$组合。(这里$\kappa_n$表示在7. 2. 1. 3 − (60)中定义的第n个Kruskal函数)。

52. [*M23*] 当随机选择嵌套括弧$a_1\cdots a_{2n}$时，求表1中量d_n的均值和方差。

53. [*M28*] 令X是扩充的二叉树的根到最左外部节点的距离。(a)当具有n个节点的所有二叉树都同样可能时，什么是X的期望值？(b)在由算法6. 2. 2T从一个随机排列$K_1\cdots K_n$构造的随机二叉查找树中，X的期望值是什么？(c)在习题31的意义下，随机退化二叉树中X的期望值是多少？(d) 在所有3种情况下，2^X的期望值是多少？

54. [*HM29*] $c_1+\cdots+c_n$的均值和方差是什么？(参见习题46)

55. [*HM33*] 试计算在习题46(a)中所有通路的总区域$C'_{pq}(1)$。

56. [*M23*] (廉佐 · 斯普鲁格诺里，1990) 试证明求和公式，对于$0\leqslant m\leqslant n$，

$$\sum_{k=0}^{m-1}C_kC_{n-1-k}=\frac{1}{2}C_n+\frac{2m-n}{2n(n+1)}\binom{2m}{m}\binom{2n-2m}{n-m}$$

57. [*M28*] 对于p=0, 1, 2, 3，以封闭形式表达求和$S_p(a, b)=\sum_{k\geqslant0}\binom{2a}{a-k}\binom{2b}{b-k}k^p$，并且使用这些公式证明(30)。

58. [*HM34*] 令t_{lmn}是当以对称顺序从0到n对树的外部节点进行编号时，外部节点m出现在层l中的n节点二叉树的个数。并令$t_{mn}=\sum_{l=1}^{n}lt_{lmn}$，使得$t_{mn}/C_n$是外部节点$m$的平均层；并令$t(w, z)$是超生成函数

$$\sum_{m,n}t_{mn}w^mz^n=(1+w)z+(3+4w+3w^2)z^2+(9+13w+13w^2+9w^3)z^3+\cdots$$

试证明$t(w, z)=(C(z)-wC(wz))/(1-w)-1+zC(z)t(w, z)+wzC(wz)t(w, z)$，并且推导对于数$t_{mn}$

39 的一个简单公式。

59. [*HM29*] 类似地，令T_{lmn}统计内部节点m出现在l层的所有n节点二叉树的棵数。试求出对于$T_{mn}=\sum_{l=1}^{n}lT_{lmn}$的一个简单公式。

▶ 60. [*M26*] (平衡串)。嵌套括弧串α是*原子的*，如果它有 (α')的形式，其中α'是嵌套的；每个嵌套串可惟一地表示作原子$\alpha_1\cdots\alpha_r$的乘积。具有相等个数的左和右括弧的一个串叫做*平衡的*；每个平衡串可以惟一地表示为$\beta_1\cdots\beta_r$，其中每个β_j是一个原子，或者是一个*协原子*(一个原子的颠倒)。一个平衡串的*不足*是其协原子长度的一半。例如，平衡串

(( ))) ((( )))))) (( ) ((( )) ((( )) (( )

有分解了的形式$\beta_1\beta_2\beta_3\beta_4\beta_5\beta_6\beta_7\beta_8=\alpha_1\alpha_2^R\alpha_3\alpha_4^R\alpha_5^R\alpha_6\alpha_7^R\alpha_8$，共有4个原子和4个协原子，

它的不足是$|\alpha_2\,\alpha_4\,\alpha_5\,\alpha_7|/2=9$。

a) 试证明一个平衡串的不足是下标k的数目，k下标表示第k个右括弧出现在第k个左括弧之前。

b) 如果$\beta_1\cdots\beta_r$是平衡的，可以通过简单地颠倒其协原子把它映射成一个嵌套的串。但以下的映射是更有趣的，因为它从无偏向的平衡串产生出无偏向的(均匀随机)嵌套串。设有s个协原子 $\beta_{i_1}=\alpha_{i_1}^R,\cdots,\beta_{i_s}=\alpha_{i_s}^R$，以(代替每个协原子，然后附加串)$\alpha'_{i_s}\cdots)\alpha'_{i_1}$，其中 $\alpha_j=(\alpha'_j)$。例如，上边的串映射成$\alpha_1(\alpha_3((\alpha_6(\alpha_8)\alpha'_7)\alpha'_5)\alpha'_4)\alpha'_2$。它刚好等于本节开始处图示的串(1)。

c) 再设计逆映射的一个算法。给定一个嵌套串$\alpha=a_1\cdots a_{2n}$和满足$0\leqslant l\leqslant n$的一个整数l，试计算对于不足l的一个平衡串$\beta=b_1\cdots b_{2n}$，对于它有 $\beta\mapsto\alpha$。不足11的什么平衡串映射到(1)?

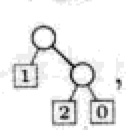
▶61. [*M26*] (莱尼循环引理。)令$b_1\,b_2\cdots b_N$是非负整数的一个串，使$f=N-b_1-b_2-\cdots-b_N>0$。

a) 试证明对于$1\leqslant j\leqslant N$，恰有循环移位$b_{j+1}\cdots b_N\,b_1\cdots b_j$的$f$满足习题20中的前根顺序的度序列性质。

b) 给定$b_1\,b_2\cdots b_N$，设计一个有效算法来确定所有这样的j。

c) 说明怎样生成有$N=n_0+\cdots+n_t$个节点的一个随机森林，恰有度数为j的n_j个节点。(例如，当$N=tn+1$，$n_0=(t-1)n+1$，$n_1=\cdots=n_{t-1}=0$和$n_t=n$时，我们得到随机n节点t叉树作为这个一般过程的一个特殊情况。)

62. [*22*] 一棵二叉树也可通过二进位串$(l_1\cdots l_n, r_1\cdots r_n)$来表示，其中$l_j$和$r_j$告知，在前根顺序节点$j$的左和右子树是否为非空。(参见定理2. 3. 1A。)试证明，如果$l_1\cdots l_n$和$r_1\cdots r_n$是满足$l_1+\cdots+l_n+r_1+\cdots+r_n=n-1$的任意二进位串，则恰有一个循环移位$(l_{j+1}\cdots l_nl_1\cdots l_j, r_{j+1}\cdots r_n r_1\cdots r_j)$产生一棵有效的二叉树表示，并且说明如何求出它。

63. [*16*] 如果勒米算法的前两次迭代已产生 ，则在下次迭代之后可能产生什么修饰过的二叉树?

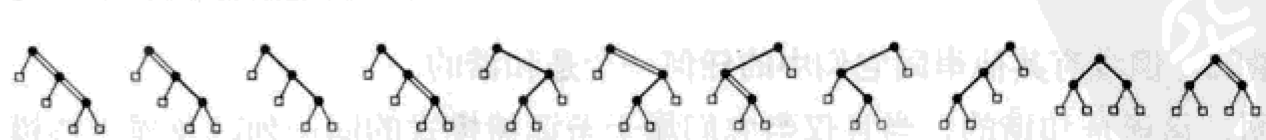
64. [*20*] 在算法R中X值的什么序列对应于(34)的修饰过的树，而且$L_0\,L_1\cdots L_{12}$的最后值是什么? 40

65. [*38*] 把勒米算法(算法R)推广到t叉树。

66. [*21*] 施罗德树是这样一棵二叉树，其中每个非空右链接以白色或黑色着色。对于小的n，n节点施罗德树的个数S_n是

n	0	1	2	3	4	5	6	7	8	9	10	11	12
S_n	1	1	3	11	45	197	903	4279	20793	103049	518859	2646723	13648869

例如，S_3=11，因为可能性为



(白链接是“空的”；外部节点也已加上。)

a) 求具有n个内部节点的施罗德树和具有n+1个叶和没有度数为1的节点的普通树之间的一个简单对应。

b) 试对于施罗德树设计一个格雷码。

67. [*M22*] 什么是对于施罗德数的生成函数$S(z)=\sum_n S_n z^n$。

68. [*10*] 什么是阶为0的圣诞树模式？

69. [*20*] 表4中阶为6和7的圣诞树模式（可能稍微掩饰）是可见的吗？

▶70. [*20*] 对于每个二进位串σ，求出定义称为配偶的另一个二进位串σ'的一个简单规则，它具有如下性质：(i) $\sigma''=\sigma$；(ii) $|\sigma'|=|\sigma|$；(iii) $\sigma\subseteq\sigma'$或者 $\sigma'\subseteq\sigma$；(iv) $\nu(\sigma)+\nu(\sigma')=|\sigma|$。

71. [*M21*] 令M_{tn}是n二进位串最大可能集合S的大小且有以下性质：如果σ和τ是满足$\sigma\subseteq\tau$ 的S成员，则$\nu(\tau)<\nu(\sigma)+t$。(因此，例如，由斯泊纳定理$M_{1n}=M_n$)。求M_{tn}的一个公式。

▶72. [*M28*] 如果你以长度为s的一行$\sigma_1\ \sigma_2\cdots\ \sigma_s$开始，反复$n$次应用增长规则(36)，你会得到多少行？

73. [*15*] 在30阶的圣诞树模式中，包含二进位串011001001000011111101101011100的行中头一个和最后一个元素是什么？

74. [*M26*] 继续上一道题，在这行之前有多少行？

▶75. [*HM23*] 令 $(r_1^{(n)},r_2^{(n)},\cdots,r_{n-1}^{(n)})$ 是n阶圣诞树模式有$n-1$项的行号；例如，表4告诉我们，$(r_1^{(8)},\cdots,r_7^{(8)})=(20,40,54,62,66,68,69)$。试求 $r_{j+1}^{(n)}-r_j^{(n)}$ 和 $\lim_{n\to\infty}r_j^{(n)}/M_n$ 的公式。

76. [*HM46*] 试研究当$n\to\infty$时圣诞树模式的极限形状。例如，它是否存在某个适当比例下的一个分形维数？

77. [*21*] 给定n，试设计一个算法生成在圣诞树模式的行中最右元素$a_1\cdots a_n$的序列。提示：这些二进位串以下列性质进行表征，即对于$0\leqslant k\leqslant n$, $a_1+\cdots+a_k\geqslant k/2$。

78. [*20*] 真或假：如果$\sigma_1\cdots\sigma_s$是圣诞树模式的一行，则 $\bar{\sigma}_s^R\cdots\bar{\sigma}_1^R$ 也是(颠倒的补的颠倒序列)。

79. [*M26*] 根据等式5. 1. 3-(12)，恰好有一个“下降”即$p_k>p_{k+1}$的排列$p_1\cdots p_n$的个数是欧几里得数 $\left\langle{n \atop 1}\right\rangle=2^n-n-1$。在底部行上边，在圣诞树模式中项的个数是相同的。

a) 通过给出在一个下降排列和未排序的二进位串之间的一一对应，试求这个巧合的一个组合说明。

b) 试说明，两个未排序的二进位串属于圣诞树模式的同一行当且仅当它们在鲁滨逊-施
41 森斯迪德对应下它们对应于定义相同P表景的排列(定理5. 1. 4A)。

80. [*30*] 我们说两个二进位串是**和谐的**，如果在子串上通过变换$010\leftrightarrow100$或$101\leftrightarrow110$，能从一个串得到另一个串。例如串

$$\begin{array}{ccccccccc} 011100 & \leftrightarrow & 011010 & \leftrightarrow & 010110 & \leftrightarrow & 010101 & \leftrightarrow & 011001 \\ & & & & \updownarrow & & \updownarrow & & \\ & & & & 100110 & \leftrightarrow & 100101 & \leftrightarrow & 101001 \leftrightarrow 110001 \end{array}$$

是互相和谐的，但没有其他串同它们中的任何一个是和谐的。

试证明，诸串是和谐的，当且仅当它们属于圣诞树模式的同一列以及属于该模式中相同长度的诸行中。

81. [*M30*] (n, n')阶的**双杂乱**是二进位串对(σ, σ')的一个家族S，其中$|\sigma|=n$和$|\sigma'|=n'$，且具有以下性质，即S的不同成员(σ, σ')和(τ, τ')，仅当$\sigma\neq\tau$和$\sigma'\neq\tau'$ 时才允许满足 $\sigma\subseteq\tau$ 和 $\sigma'\subseteq\tau'$。

使用圣诞树模式证明S至多包含$M_{n+n'}$个串对。

▶82. [*M26*] 令$E(f)$是算法H计算函数f的执行次数。

a) 证明$M_n \leqslant E(f) \leqslant M_{n+1}$，且当$f$是常数时，等式成立。

b) 在使$E(f)=M_n$的所有f当中，哪一个使$\Sigma_\sigma f(\sigma)$成为极小？

c) 在使$E(f)=M_{n+1}$的所有f当中，哪一个使$\Sigma_\sigma f(\sigma)$成为极大？

83. [*M20*] (乔 · 汉塞尔) 试证明至多有3^{M_n}个带有n个布尔变量的单调布尔函数$f(x_1,\cdots,x_n)$。

▶84. [*HM27*] (丹 · 克雷特曼)令A是一个实数的$m\times n$矩阵，其中每列v有长度$\|v\|\geqslant 1$，并令b是一个m维列向量。试证明至多有M_n个列向量$x=(a_1,\cdots,a_n)^T$，其分量$a_j=0$或1，满足$\|Ax-b\|<1/2$。提示：使用类似于圣诞树模式的一个构造。

85. [*HM35*] (菲律比 · 戈勒令V是包含在所有实数n维向量集合中，而不含单位向量(1, 0, ⋯, 0), (0, 1, 0, ⋯, 0), ⋯, (0, 0,⋯, 1)的任何向量空间。试证明，V至多包含M_n个分量为全0或全1的向量：而且，上界M_n是可达到的。

86. [*15*] 如果把(2)当作一个有向原森林而不是一个有序森林，问什么规范森林对应于它？试通过森林的层次代码$c_1\cdots c_{15}$以及它的父指针$p_1\cdots p_{15}$来描述之。

87. [*M20*] 令F是一个有序森林，前根顺序的第k个节点出现在层c_k，并有父亲p_k，其中$p_k=0$如果这个节点是一个根。

a) 对于$1\leqslant k\leqslant n$，有多少森林满足条件$c_k=p_k$？

b) 假设F和F'分别有层次码$c_1\cdots c_n$和$c'_1\cdots c'_n$，以及父链接$p_1\cdots p_n$和$p'_1\cdots p'_n$。试证明，在词典顺序下，$c_1\cdots c_n\leqslant c'_1\cdots c'_n$当且仅当$p_1\cdots p_n\leqslant p'_1\cdots p'_n$。

88. [*M20*] 试分析算法O：步骤O4执行的频率如何？在步骤O5中p_k改变的总次数是多少？

89. [*M46*] 步骤O5置$p_k\leftarrow p_j$的频率如何？

▶90. [*M27*] 如果$p_1\cdots p_n$是一个有向森林的父指针的一个规范序列，则具有顶点$\{0,1,\cdots,n\}$和边$\{k — p_k \mid 1\leqslant k\leqslant n\}$的图是一棵自由树，即不含环的一个连通图。(参见定理2.3.4.1A)。反之，每棵自由树在这种方式下至少对应一个有向森林。但父指针011和000两者都产生相同的自由树 ≻—；类似地，012和010都产生 •—•—• 。 42

本题的目的是进一步限制序列$p_1\cdots p_n$，使得每棵自由树恰得到一次。在2. 3. 4. 4-(9)中我们证明了，通过证明自由树总是至少有一个矩心，则$n+1$个顶点的结构不同的自由树的个数有一个相当简单的生成函数。

a) 试证明一个规范的n节点森林对应于有一个矩心的自由树当且仅当在森林中没有多于$\lfloor n/2\rfloor$个节点的树。

b) 试修改算法O使得它生成满足(a)的所有序列$p_1\cdots p_n$。

c) 试说明如何求出有两个矩心的自由树的所有$p_1\cdots p_n$。

91. [*M37*] (尼珍休斯和威尔弗。)试证明一棵随机有向树可以通过类似于习题7. 2. 1. 4-47的随机分划算法的过程生成。

92. [*15*] 在共同有$n-2$条边的意义下，由算法S访问的头一棵和最后一棵生成树是否相邻？

93. [*20*] 当算法S终止时，是否把图恢复成它原来的状态？

94. [*22*] 算法S需要通过在步骤S1中寻找一棵初始的生成树来启动进展。试说明，应如何完成这个任务。

95. [*26*] 通过实现步骤S8中的桥测试来完成算法S。

▶96. [*28*] 当给定的图只不过是(a)长度为$n-1$的一条通路P_n；(*b*)长度为n的一个环C_n时，试分析算法S的近似运行时间。

97. [*15*] (48)是一个串-并行图吗?

98. [*16*] 如果把A取成串行时，什么串-并行图对应于(53)?

▶99. [*30*] 考虑像在(53)中的由树表示连同满足(55)的节点值的串-并行图。按照在根p处v_p是1或0，这些值定义一棵生成树或一棵接近树。试证明，下列方法将生成这个根的所有其他配置：

i) 以所有非省事的节点主动开始，而其他节点为被动。

ii) 以前根顺序选择最右活动节点p；如果所有节点为被动的，则终止。

iii) 改变$d_p \leftarrow r_{d_p}$，更新树中所有的值，访问新配置。

iv) 激活p右边的所有非省事节点。

v) 如果自p最后变成活动以来d_p已经遍历p的所有儿子，则使节点p成为被动的。返回(ii)。

还要说明如何有效地实施这些步骤。提示：为实现步骤(v)，引进一个指针z_p；当d_p变成等于z_p时，使节点p成为被动的，而且在这样的时刻也把z_p还原成d_p的前一个值。为实现步骤(ii)和(iv)，使用类似于在算法7.2.1.1L和7.2.1.1K中那些焦点指针f_p。

100. [*40*] 通过组合算法S和习题99的思想，实现转动门址生成所有生成树的正文中的“算法S′”。

101. [*46*] 有无简单的转动门方法列出完全图K_n的所有n^{n-2}棵生成树？（由算法S产生的序十分复杂。）

102. [*46*] n个顶点的有向图D的有向生成树，也叫做“生成树形图”，是包含$n-1$个弧的D的一棵有向子树。矩阵树定理(习题2.3.4.2-19)告诉我们，有一个给定根的有向子树可以通过计算一个$(n-1)\times(n-1)$的行列式容易地计算出来。

43 总是删去一个弧而以另一个弧代替，这些有向子树能以转动门顺序列出吗？

▶103. [*HM39*] (沙堆。)考虑顶点$V_0, V_1, \cdots, V_n$和从V_i到V_j的弧e_{ij}(其中$e_{ii}=0$)的有向图D。假定D至少有以V_0为根的一棵有向生成树。这个假定意味着，如果我们适当地对顶点编号，对于$1\leqslant i\leqslant n$，我们有$e_{i0}+\cdots+e_{i(i-1)}>0$。令$d_i=e_{i0}+\cdots+e_{in}$是$V_i$的总出度。对于$0\leqslant i\leqslant n$，在顶点$V_i$放上$x_i$个沙粒，并且玩下列游戏：对于任何$i\geqslant 1$，如果$x_i\geqslant d_i$，则$x_i$减$d_i$，对于所有$j\neq i$，置$x_j\leftarrow x_j+e_{ij}$。(换言之，每当可能时就传送一粒沙到它的每条出边，但当$i=0$时除外。这个操作叫做“颠覆”V_i，而一序列的颠覆称为一个“雪崩”。顶点V_0是特殊的，代替颠覆，它收集实际上离开系统的沙粒。)继续进行，直到对于$1\leqslant i\leqslant n$，$x_i<d_i$为止。这样的状态$x=(x_1, x_2, \cdots, x_n)$称为**稳定的**。

a) 试证明，在有限次颠覆之后，每一个雪崩在一个稳定状态下终止。而且，最终状态仅依赖初始状态，而不依赖于实施颠覆的顺序。

b) 设$\sigma(x)$是由初始状态x得到的稳定状态。一个稳定状态叫做**递归状态**，如果对于$1\leqslant i\leqslant n$，满足$x_i\geqslant d_i$的某个x，它是$\sigma(x)$。(在随机重复引进新的沙堆之后，递归状态对应于已经增长了长时间的沙堆。)求当$n=4$的特殊情况下和当D的仅有弧为

$$V_1\to V_0,\ V_1\to V_2,\ V_2\to V_0,\ V_2\to V_1,\ V_3\to V_0,\ V_3\to V_4,\ V_4\to V_0,\ V_4\to V_3$$

时的递归状态。

PDG

c) 令$d=(d_1,\cdots,d_n)$。证明x是递归的当且仅当$x=\sigma(x+t)$，其中t是向量$d-\sigma(d)$。

d) 对于$1\leqslant i\leqslant n$，令a_i为向量$(-e_{i1},\cdots,-e_{i(i-1)},d_i,-e_{i(i+1)},\cdots,-e_{in})$。因此，颠覆$V_i$对应于把状态向量$x=(x_1,\cdots,x_n)$改变成$x-a_i$。说两个状态$x$和$x'$是同余的，写成$x\equiv x'$，如果对于某些整数$m_1,\cdots,m_n$，$x-x'=m_1a_1+\cdots+m_na_n$。试证明，恰好有和以$V_0$为根的$D$中有向生成树一样多的同余状态等价类。提示：参看矩阵树定理，即习题2. 3. 4. 2-19。

e) 如果$x\equiv x'$，而且，若x和x'两者都是递归的，证明$x=x'$。

f) 试证明，每个同余类包含惟一的递归状态。

g) 如果在每个顶点的入度等于它的出度的意义下，D是平衡的，试证明x是递归的当且仅当$x=\sigma(x+a)$，其中$a=(e_{01},\cdots,e_{0n})$。

h) 当D是一个具有n条幅的“轮子”时，试说明这些概念：设有$3n$条弧，即对于$1\leqslant j\leqslant n$，$V_j\to V_0$和$V_j\leftrightarrow V_{j+1}$并且认为$V_{n+1}$等于$V_1$。试求这个有向图的有向生成树和它的沙堆的递归状态之间的一一对应。

i) 类似地，当D是$n+1$顶点的完全图时，即当对于$0\leqslant i,\ j\leqslant n$，$e_{ij}=[i\neq j]$时，试分析递归的沙堆。提示：参见习题6.4-31。

▶104. [*HM21*] 如果G是n个顶点$\{V_1,\cdots,V_n\}$的图，且V_i和V_j之间有边e_{ij}，令$C(G)$是具有项$c_{ij}=-e_{ij}+\delta_{ij}d_i$的矩阵，其中$d_i=e_{i1}+\cdots+e_{in}$是$V_i$的度。我们说$G$的外表是$C(G)$的特征值，即方程$\det(\alpha I-C(G))=0$的根$\alpha_0,\cdots,\alpha_{n-1}$。由于$C(G)$是对称矩阵，它的特征值是实数，我们可以假设$\alpha_0\leqslant\alpha_1\leqslant\cdots\leqslant\alpha_{n-1}$。

a) 试证明$\alpha_0=0$；

b) 试证明G恰有$c(G)=\alpha_1\cdots\alpha_{n-1}/n$棵生成树； 44

c) 什么是完全图K_n的外表？

105. [*HM37*] 继续习题104，我们希望证明，当G已从其外表已知的其他图构造出来时，通常有容易的方法来确定G的外表。假设G'有外表$\alpha'_0,\cdots,\alpha'_{n'-1}$，而$G''$有外表$\alpha''_0,\cdots,\alpha''_{n''-1}$，在下列情况下$G$的外表是什么？

a) $G=\overline{G'}$是G'的补。(在此情况下，假定$e'_{ij}\leqslant[i\neq j]$。

b) $G=G'+G''$是G'和G''的和(并列)。

c) $G=G'\mp G''$是G'和G''协和(并)。

d) $G=G'\times G''$是G'和G''的笛卡儿乘积。

e) $G=L(G')$是G'的线图，当G'是度数d'的一个正则图时(即当G'的所有顶点恰有d'个邻居而没有自循环时)。

f) $G=G'\Diamond G''$是G'和G''的直接积(与)，当G'是度数为d'的正则图，而G''是度数d''的正则图时。

g) $G=G'\otimes G''$是正则图G'和G''的强乘积。

▶106. [*HM37*] 求在(a)一个$m\times n$的栅格$P_m\times P_n$，(b)一个$m\times n$的圆柱体$P_m\times C_n$；(c)一个$m\times n$圆环体$C_m\times C_n$中生成树的个数。为什么这些数趋向于只有小的质因子。提示：试证明，借助于数$\sigma_{kn}=4\sin^2\dfrac{k\pi}{2n}$可表达$P_n$和$C_n$的外表。

107. [*M24*] 试确定有$n\leqslant 5$个顶点、没有自循环或并行边的所有连通图的外表。

108. [*HM40*] 把习题104～106的结果推广到有向图。

109. [*M46*] 试求对于(57)是在n立方体中生成树的个数这一事实的组合说明。

▶110. [*M27*] 试证明，如果G是没有自循环的连通多图，则它有

$$c(G) > \sqrt{(d_1-1)\cdots(d_n-1)}$$

棵生成树，其中d_j是顶点j的度。

▶111. [*05*] 以后前根顺序列出树(58)的节点。

112. [*15*] 如果一个森林的节点p在前后根顺序下居于节点q之前，而在后前根顺序下居于它之后，对于p和q你能说些什么？

▶113. [*20*] 一个森林F的前后根顺序和后前根顺序如何同共轭森林F^R的前后根顺序与后前根顺序相关联？(参见习题13)。

114. [*15*] 如果要使用算法Q在前后根顺序下遍历整个森林，应如何开始这一过程？

115. [*20*] 试分析算法Q：在对一个森林进行完全遍历期间，执行每一步的频率是多少？

▶116. [*28*] 对于$1\leqslant k\leqslant n$，如果在前后根顺序下以1至n对森林F的节点进行标号，我们说节点k是*幸运的*，如果在F中它同节点$k+1$相邻的话，说节点k是*不幸运的*，如果它们相距3步，否则它是*通常的*。在此定义下，节点$n+1$被认为是每个根的父亲的一个虚拟的超根。

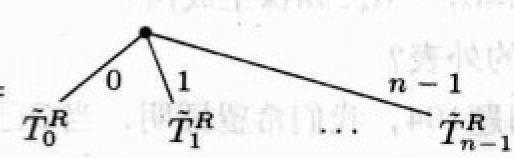
a) 试证明幸运节点仅在偶数编号层出现。不幸运节点仅在奇数编号层出现。

b) 试证明幸运节点的数目恰好比不幸运节点的数目大1，除非$n=0$。

45 117. [*21*] 有多少n节点森林不含不幸运节点？

118. [*M28*] 在(a)具有$(t^k-1)/(t-1)$个内部节点的完全t叉树中(b)在具有$F_{k+1}-1$内部节点的k阶斐波那契树中，存在多少个幸运节点？(参看2.3.4.5-(6)和6.2.1节中的图8。)

119. [*21*] 根据规则，对于$n>0$，

$\tilde{T}_0 = \bullet$，　　$\tilde{T}_n =$  对于$n>0$

递归地定义n阶的*扭曲的二项式树* $\tilde{T}_n$。(同7.2.1.3-(21)作比较，我们在交替层上颠倒儿子的次序。)试证明$\tilde{T}_n$的前后根顺序遍历有同格雷二进码的简单联系。

120. [*22*] 真或假：一个图的平方是哈密顿的，如果这个图连通且无桥。

121. [*M32*] (弗·纽曼，1964。)一个图G的*推导*是通过从中删去度数为1的所有顶点以及连接它们的边得到的图$G^{(\prime)}$。试证明，当T是一棵自由树时，它的平方T^2包含哈密顿通路当且仅当它的推导没有度数大于4的顶点，而且两个附加的条件成立：

i) 在$T^{(\prime)}$中度数为3或4的所有顶点位于一条通路上。

ii) 在$T^{(\prime)}$中度数为4的任何两个顶点之间，至少有一个顶点，它在T中度数为2。

▶122. [*31*] (杜德尼数字世纪难题。)通过把算术操作符和也许可能还有括弧插入序列123456789中，有许多奇怪的方式来得到数100。例如，

$$100 = 1+2\times3+4\times5-6+7+8\times9 = (1+2-3-4)\times(5-6-7-8-9)$$
$$= ((1/((2+3)/4-5+6))\times7+8)\times9$$

a) 100可能有多少这样的表示？为使这个问题精确化，通过结合律和其他代数性质，假定表达式按照以下语法以正则形式写成：

〈表达式〉→〈数〉|〈和〉|〈积〉|〈商〉

〈和〉→〈项〉+〈项〉|〈项〉−〈项〉|〈和〉+〈项〉|〈和〉−〈项〉

〈项〉→〈数〉|〈积〉|〈商〉

〈积〉→〈因子〉×〈因子〉|〈积〉×〈因子〉|(〈商〉)×〈因子〉

〈商〉→〈因子〉/〈因子〉|〈积〉/〈因子〉|(〈商〉)/〈因子〉

〈因子〉→〈数〉|(〈和〉)

〈数〉→〈数字〉

所用数字必须是1顺序到9。

b) 通过允许多个数字的数并用语法

〈数〉→〈数字〉|〈数〉〈数字〉

来推广问题(a)，例如，100=(1/(2−3+4))×567−89。这样的最短表示是什么？最长的是什么？

c) 通过也允许小数点来推广问题(b)：

〈数〉→〈数字串〉|.〈数字串〉

〈数字串〉→〈数字〉|〈数字串〉〈数字〉 46

例如，令人惊讶的是，100=(0. 1−2−34×0. 5)/(0. 6−0. 789)。

123. [*21*] 继续上道题，不能使用约定(a)(b)(c)表示的最小正整数是什么？

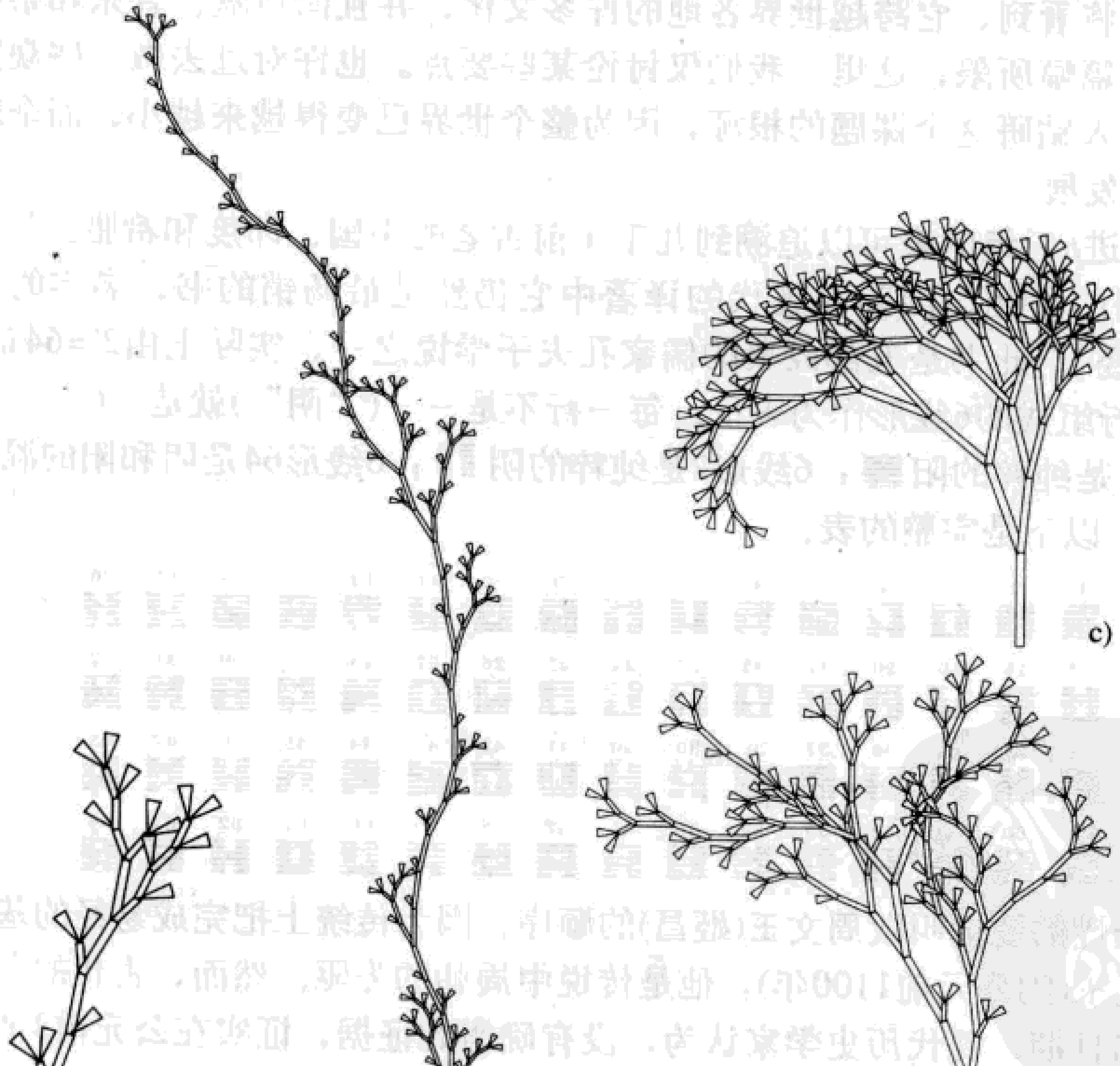


图42 二叉树的“有机”图示

▶124. [*40*] 在来自于自然的简单模型的启发下尝试画扩充的二叉树的一些方法。比如，

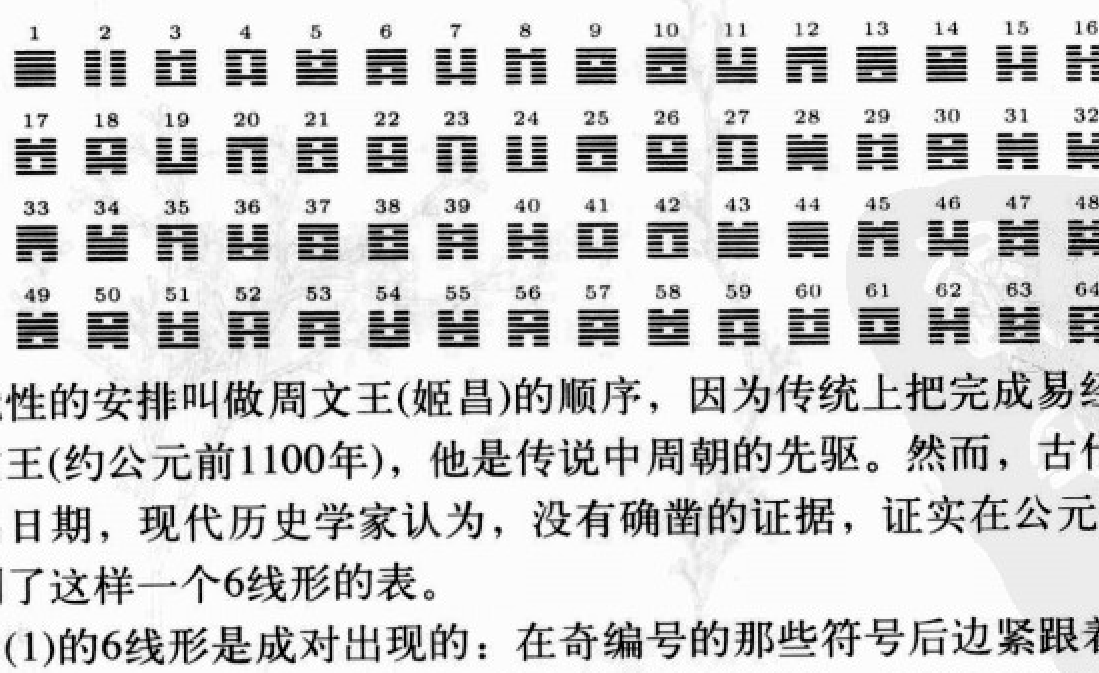
把值$v(x)$赋给每个节点x，叫做霍顿–斯特拉勒数，如下：每个外部(叶)节点有$v(x)=0$；有儿子(l, r)的一个内部节点有$v(x)=\max(v(l), v(r))+[v(l)=v(r)]$。对于某些函数$h$,$w$和$\theta$，由内部节点$x$到它的父亲的边可以画作高$h(v(x))$和宽$w(v(x))$的一个矩形，而且有儿子$(l, r)$的边的矩形可以通过角$\theta(v(l(x)),v(r(x)))$，$-\theta(v(r(x)), v(l(x)))$而偏离。图42中的例子对于$0\leqslant k<j$时，显示当我们选择$w(k)=3+k$，$h(k)=18k$，$\theta(k, k)=30^\circ$，$\theta(j, k)=((k+1)/j)\times 20^\circ$的典型结果，以及对于$0\leqslant j<k$，$\theta(j, k)=((k-j)/k)\times 30^\circ$的典型结果；根出现在底部。图42a是二叉树(4)；图42b是由算法R生成的一棵随机的100节点的树；图42c是11阶的斐波那契树，它有143个节点；图42d是一棵随机的100节点的二叉查找树。(图42b、c、d中的树显然属于
47 不同品种。)

(本课题)几乎同人类思想可加以应用的每种有用知识都有关系。
——詹姆斯·伯努利，*Ars Conjectandi* (1713)

7.2.1.7 历史和进一步的参考文献

组合模式生成的早期工作可以追溯到文明刚开始成形的时期。这个故事十分有趣，我们将看到，它跨越世界各地的许多文化，并且同诗歌、音乐和宗教紧密相联。由于篇幅所限，这里，我们仅讨论某些要点。也许对过去做一些观察，将促使读者深入钻研这个课题的根源，因为整个世界已变得越来越小，而全球的学术界在继续发展。

类似二进n元组的表可以追溯到几千年前古老的中国、印度和希腊。最著名的来源是中国的易经，因为在现代的译著中它仍然是最畅销的书，名字的含义是“变化的圣经”。此书是五部经典的儒家孔夫子学说之一，实际上由$2^6=64$章组成。每一章由6行组成的6线形作为表征。每一行不是– –(“阴”)就是—(“阳”)。例如，6线形1是纯粹的阳䷀；6线形2是纯粹的阴䷁；6线形64是阴和阳的混合，而阳在顶䷿。以下是完整的表：


(1)

这64个可能性的安排叫做周文王(姬昌)的顺序，因为传统上把完成易经的基本内容归功于周文王(约公元前1100年)，他是传说中周朝的先驱。然而，古代的书籍很难可靠地定出日期，现代历史学家认为，没有确凿的证据，证实在公元前3世纪，到底是谁编制了这样一个6线形的表。

注意，(1)的6线形是成对出现的：在奇编号的那些符号后边紧跟着其上下对称颠倒的符号，除了上下颠倒时符号没有变化的情况例外。而8个对称的图和它们的补成对出现 ($1=\overline{2}, 27=\overline{28}, 29=\overline{30}, 61=\overline{62}$) 。6线形由两个3线形组成，这3线形表示4个基本元素：天(☰)、地(☷)、火(☲)和水(☵)，而且这4个元素放置在适当位置。否则，这个安排看上去好像是随机的，似乎是在数学上缺乏训练的人来在列

出不同的可能性直到不能再有任何可能性为止。在一些对之间确实存在一些有趣的模式，但是不会像π的数字那样具有偶然性（参见3.3-(1)）。 48

阴和阳代表了自然的基本力量的互补方面，总紧密相连，总处于变化中。易经有些类似于百科全书，其中6线形用作一个索引，索引关于基本概念的累积的智慧，像乾(䷀)、坤(䷁)、谦(䷎)、兑(䷹)、同人(䷌)、遯(䷠)、泰(䷊)、讼(䷅)、师(䷆)、蛊(䷑)、蒙(䷃)、贲(䷕)等等。人们可随机地选择一对6线形组合，通过某种方法改变成第二种组合，比如说，以1/4的概率独立地改变每个阴成为阳(或反过来)，这种方法能产生出4096种结果，用来表达世间万物。与马尔可夫过程一样，其变化之多也许能给出生命的意义。

大约在公元1060年，邵雍以严格的逻辑方法对6线形进行了重新排序。他的排序方法是按词典顺序从䷁到䷖，到䷇，到䷓，到䷏，…，到䷪，到䷀。(从底向上读每个6线形)。这比起周文王的排序方法更为用户友好，因为现在可以很快地找出一个随机模式来。当戈·威·莱布尼茨于1702年了解了这个6线形序列时，他匆匆作出结论说，中国数学家已经熟悉二进制算术[参见弗朗克·斯威特兹(*Mathematics Magazine* **76**(2003),276-291。关于易经进一步的细节，可以参见李约瑟的*Science and Civilisation in China* **2**(剑桥大学出版社，1956)，304-345，或理·约·林，*The Classic of Changes*(纽约：哥伦比亚大学出版社，1994)。]

另一位古代中国哲学家扬雄提出基于81个三进制的四线形体系，以代替64个二进制的6线形。他的《太玄》写于公元前2世纪，最近由迈克尔·尼兰译成英文(阿尔巴尼，纽约：1993)。扬雄描述了一个完备的，分层的三叉树结构，其中有3个区域，每个区域有3个省，每个省中有3个部门，在每个部门有3个家庭，以及每个家庭有9个叫做“评价”的9首短诗。因此，一共有729个评价——组成一年中每天几乎恰好有2个评价。他的四线形从上往下严格按照词典顺序排列：𝌆,𝌇,𝌈,𝌉,𝌊,𝌋,𝌌,…,𝍖。事实上，如同尼兰的书第28页上所说明的，扬雄介绍了计算每个四线形的阶的一个简单方法，就好像使用一个3进制系统一样。而邵雍的二进制6线形系统并未受其影响，尽管邵雍生活在1000多年之后。

印度的韵律。古代印度的博学大师们也研究了类似二进制n元组的内容，他们研究宗教的吠陀赞美诗的诗音步。在梵语中的音节是短的(।)或是长的(ऽ)，而对于音节模式的研究称为“韵律”。现代作者使用符号⏑和—来代替।和ऽ。一个典型的吠陀诗句由四行组成，对于某个$n \geq 8$，每行有n个音节；因此，韵律学家寻求一个方法对所有2^n种可能性进行分类。由平加拉写成于公元400年以前或许更早(确切的日期十分不确定)的经典著作*Chandaḥśāstra*。人们通过它能容易地找出诸।和诸—的任何给定模式的下标k，以及给定k求第k个模式的过程。换言之，平加拉说明怎样对给定的模式安排阶以及对给定的下标去掉阶；因此，他的工作超出了 49
扬雄的工作，扬雄只考虑了排阶而未考虑去阶。平加拉的方法也同指数有关，如同我们早在算法4.6.3A相联系时已经说明过的那样。

下一个重要贡献是由叫做克达拉的韵律学家在他的著作(*Vṛttaratnākara*.)中所采取的，尽管它是在8世纪才写成的。克达拉给出了逐步过程，列出所有n元组

从———...—到⏑——...—，到—⏑—...—，到⏑⏑—...—，到——⏑...—，到⏑—⏑...—，到…到⏑⏑⏑...⏑，这实质上是在二进制情况下的算法7.2.1.1M。他的方法可能是对于组合序列生成的头一个明确的算法。[参见巴 · 范 · 奴廷，*J. Indian Philos.* **21** (1993)，31-50。]

诗步也可当作韵律，每个⏑是一拍，每个—是两拍。一个n音节模式在n和$2n$拍之间，但是，适合于行进或跳舞的音乐韵律一般基于固定数目的节拍。因此，自然考虑对于固定的m，恰好有m拍的所有⏑和—序列的集合。这样的模式与现在叫做长度m的莫斯(Morse)码序列相似，而且由习题4.5.3-32我们知道，它们共有F_{m+1}。例如，当m=7时，21个序列是

⏑———，—⏑——，⏑⏑⏑——，——⏑—，⏑⏑—⏑—，
⏑—⏑⏑—，—⏑⏑⏑—，⏑⏑⏑⏑⏑—，———⏑，
⏑⏑——⏑，⏑—⏑—⏑，—⏑⏑—⏑，⏑⏑⏑⏑—⏑，　　(2)
⏑——⏑⏑，—⏑—⏑⏑，⏑⏑⏑—⏑⏑，——⏑⏑⏑，
⏑⏑—⏑⏑⏑，⏑—⏑⏑⏑⏑，—⏑⏑⏑⏑⏑，⏑⏑⏑⏑⏑⏑⏑.

这样一来，印度的韵律学家所做的工作类似于斐波那契序列，如在1.2.8节介绍的那样。

而且，*Prākṛta Paiṅgala*（约公元1320年）的匿名作者发现了关于m拍基律的排阶和除阶的优雅算法。为求出第k个模式，人们写下m个⏑开始，然后表达差$d=F_{m+1}-k$作为斐波那契数$F_{j_1}+\cdots F_{j_t}$之和；这里F_{j_1}是$\leqslant d$的最大斐波那契数，F_{j_2}是$\leqslant d-F_{j_1}$的最大斐波那契数，等等；继续进行直到余数为零为止。然后，对于$j=j_1, \cdots, j_t$，拍子$j-1$和j从⏑⏑改变成—。例如，为得到(2)的第5个元素，计算$21-5=16=13+3=F_7+F_4$；答案为⏑⏑—⏑—。

一些年后，纳拉燕纳 · 潘迪塔处理了寻找其$\leqslant q$的部分的所有m组成的一般问题，其中q是任意给定的正整数。结果，他发现第q阶的斐波那契序列5. 4. 2-(4)，600年后它注定被使用于多阶段排序中了。他还建立了对应的排阶和去阶算法。[参见帕尔玛南德 · 辛格，*Historia Mathematica* **12** (1985)，229-244以及习题16。]

平加拉对所有三音节诗步给了特殊的码名，

———＝म (m)，　　——⏑＝त (t)，
⏑——＝य (y)，　　⏑—⏑＝ज (j)，
—⏑—＝र (r)，　　—⏑⏑＝भ (bh)，　　(3)
50 ⏑⏑—＝स (s)，　　⏑⏑⏑＝न (n)，

梵语的学生们自那时以来就期望记住它们。通过发明无意义的词 *yamātārājabhānasalagām*（यमाताराजभानसलगाम्），有人很久以前设计了记住这些码的机智方法；关键点在于这个词的十个音节可以写成

ya mā tā rā ja bhā na sa la gām　　(4)
⏑ — — — ⏑ — ⏑ ⏑ ⏑ —

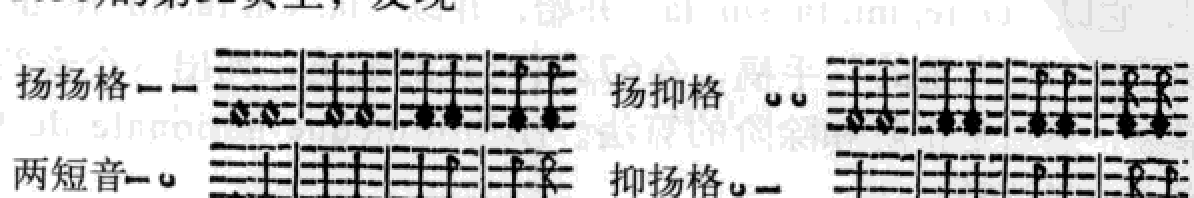
而且每三个音节模式恰出现在它的码名之后。yamā…lagām的起源不详，但苏巴斯 · 卡克*Indian J. History of Science* **35** (2000)，123-127]已经至少回溯到查 · 菲 · 布朗*Sanskrit Prosody* (1869)，28页。因此，它有资格成为类似于二进制n元组编码的最早的“德 · 布鲁因循环”。

同一时期在欧洲。在欧洲也有类似的发现，古典希腊的诗基于称为“诗步呎”的短和/或长的音节，类似于音乐的拍，每个基本呎的类型具有一个希腊名。例如，两个短的音节“◡◡”称为*pyrrhic*(两个短音节或非重读音节——译者注)，而两个长的音节“——”称为一个*spondee*(扬扬格)，因为这些音律分别用于战争(πυρρίχη)的歌中或和平(σπονδαί)的歌中。诗步呎的希腊名很快传播到拉丁语中，而且最后传播到包括英语在内的现代语言中。

◡ 强音部	◡◡◡◡ 四短音节格	
— 强声部	◡◡◡— 四音节的音步。一个是长音节，三个是短音节	
	◡◡—◡ 四音节的音步。两短音节后随一长音节，再一短音节	
◡◡ 抑抑格	◡◡—— 小爱奥尼音步	
◡— 抑扬格	◡—◡◡ 四音节的音步。长音节在第二个音节中	
—◡ 扬抑格	◡—◡— 由两个抑扬格组成的诗的韵	
—— 扬扬格	◡——◡ 抑扬格的韵脚后接扬抑格韵脚	
	◡——— 头一个三长一短或三扬一抑组成的音步	
◡◡◡ 三短节音步	—◡◡◡ 头一个三短一长音节的音步	(5)
◡◡— 抑抑扬格	—◡◡— 长短短长的音步	
◡—◡ 抑扬抑音步。一个三音节的音节	—◡—◡ 双扬抑格	
◡—— 巴克斯音步	—◡—— 第二个由三长一短音节组成的音步	
—◡◡ 扬抑抑格	——◡◡ 大爱奥尼音步	
—◡— 扬抑扬音步	——◡— 第三个三长一短音步	
——◡ 扬扬抑格	———◡ 第四个三长一短音步	
——— 莫洛西亚音步	———— 双扬扬格	

另外的名称，像“choree”(扬抑格)代替“trochee”(扬抑格)，或“cretic”(扬抑扬格)代替“amphimacer”(扬抑扬)，也很通用。而且，在狄俄墨得斯写拉丁文法(大约公元375年)时，32个五音节呎的每一个至少具有一个名字。狄俄墨得斯还指出互补模式之间的关系。例如，三短音节和莫洛西亚音步(三长音节)是“难以相处的”，抑扬抑音步和扬抑扬音步也一样。但他也认为扬抑抑格是抑抑扬格的反面，以及巴克斯音步(抑扬扬)是扬扬抑格的反面，尽管扬扬抑格字面的含义实际上是“颠倒的巴克斯音步”。希腊的韵律学家没有列出个别韵律可能性的标准顺序，因 51
而名字的形式很显然没有把它同二进制系统联系起来。[参见海·基尔，*Grammatici Latini* **1**(1857)，474-482；威·冯·克里斯特，*Metrik der Griechen und Römer* (1879)，78-79]。

由阿里斯托泽奴斯完成的音律基础(*Elements of Rhythm*)(约公元前325年)一书保存下来的一些篇章表明，音乐中也使用同样的术语。确实，在文艺复兴之后产生了同样的艺术。例如，我们在阿塔纳西尤斯·克彻尔的*Masurgia Universalis* **2**(罗马：1650)的第32页上，发现



而且，克彻尔继续描述了(5)的所有3音步和4音步的音律。

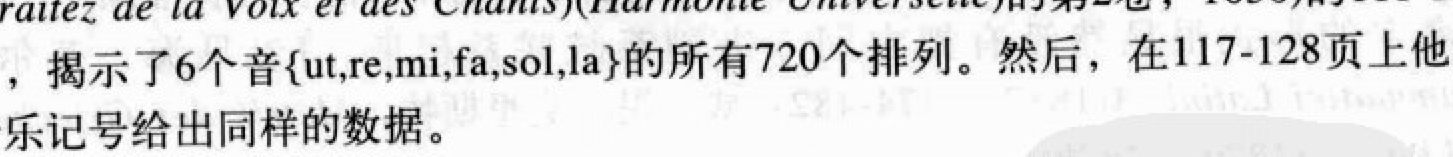
早期的排列表。我们在5.1.2节中已经追溯了计算排列的一些公式的历史。但

是，在发现了公式$n!$之后的数百年，非平凡的排列表才得以发表。目前所知的第一个这样的表是意大利物理学家沙比泰·敦诺罗在公元946年写成的犹太神秘哲学的*Sefer Yetzirah*评论中编制的。表1列出了他对于$n=5$的表，如同随后在华沙(1884年)所印出的那样。(表中的希伯莱文是以传统用于注解的犹太法学博士论文的字型打印的。注意，当字母 מ 出现在一个字的左端时，改变它的形状成为 ם 。)敦诺罗继续列出6个字母的词 םכתכנו 的120个排列，它们全都以币符号(ם)开始；然后他把其他5个字母的每一个放在前端，就可以得到另外120个排列，共有720个排列。他的表涉及了6个排列的分组，但是杂乱的方式导致他出错(参见习题4)。尽管他知道应当有多少个排列，而且知道以一个给定字母开始应该有多少个，但他最终并没有给出生成它们的一个算法。

表1　中世纪的一个排列表

דכריס, דכירס, דכימר, דכמיר, דכמרי, דכרמי, דרכיס, דרכמי, דרימכ, דריכס, דרמכי, דרמיכ,
דיכרס, דירכס, דירמכ, דיכמר, דימרכ, דימכר, דמכיר, דמכרי, דמרכי, דמריכ, דמיכר, דמירכ,
כדימר, כדירס, כדרמי, כדריס, כדמרי, כדמיר, כירמד, כירדס, כימדר, כימרד, כידמר,
כמדרי, כמדיר, כמריד, כמרדי, כמדיר, כמירד, כדמיר, כדמרי, כדרמי, כדריס, כדימר, כדירס,
רימדכ, רימכד, רידכס, רידמכ, ריכדס, ריכמד, רדמכי, רדכמי, רדיכס, רדימכ, רדכיס, רדמיכ,
רכמדי, רכדמי, רכדיס, רכידס, רכימד, רכמיד, רמיכד, רמידכ, רמדיכ, רמכיד, רמכדי, רמדכי,
ימדרכ, ימרדכ, ימכדר, ימכרד, ימרכד, ימדכר, ידכרס, ידרכס, ידכמר, ידמכר, ידרמכ, ידמרכ,
יכדרס, יכדמר, יכרדס, יכרמד, יכמדר, יכמרד, ירדכס, ירדמכ, ירמכד, ירכמד, ירכדס, ירדמכ,
מדכרי, מדרכי, מדכיר, מדיכר, מדירכ, מדריכ, מכדיר, מכידר, מכריד, מכירד, מכרדי, מכדרי,
מרדכי, מרכדי, מרדיכ, מרכיד, מריכד, מרידכ, מידכר, מידרכ, מיכרד, מיכדר, מירדכ, מירכד

{a, b, c, d, e, f}的所有720个排列的完整表出现在泽里米亚斯·德里塞尔的*Orbis Phaëthon*的668-671页上(慕尼黑：1629；也出现在1631年的戈罗格尼(Cologne)版本的526-531页上)。他提供它作为以下事实的证明，即某人有6个客人，
52 在一年的360天时间里，他每天可以不同的坐次安置他们的午餐和晚餐，因为每年有5天是复活节前一周的五天禁食。不久之后，马林·梅森在关于歌声的论著(*Traitez de la Voix et des Chants*)(*Harmonie Universelle*)的第2卷，1636)的111-115页，揭示了6个音{ut,re,mi,fa,sol,la}的所有720个排列。然后，在117-128页上他以音乐记号给出同样的数据。

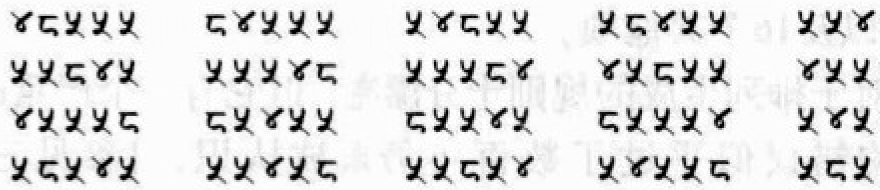


德里塞尔表按列的词典顺序组织，而梅森表相对于ut<re<mi<fa<sol<la的顺序按词典顺序排列，它以“ut, re, mi, fa, sol, la”开始，并以“la, sol, fa, mi, re, ut”结束。梅森还编制了一个“大量的”手稿，在672页对开的纸上，列出八个音符的所有40 320个排列，然后是排阶和除阶的算法。[Bibliothèque nationale de France, Fonds Français，24256号。]

我们在7.2.1.2节中看到，平易改动的重要思想即算法7.2.1.2P，是一些年后才在英格兰发明的。

列出有重复元素的多重集合的所有排列的方法经常为早期的作者们所误解。

例如，当巴斯卡拉在*Lilāvati*(大约1150年)的271节展示了{4, 5, 5, 5, 8}的排列时，他以下列顺序列出它们：


(6)

当列出拉丁名IESVS的六十个颠倒字母顺序时，梅森在他的书的第131页上使用了更敏感的但不完全系统的顺序。当阿塔纳西尤斯·克彻尔要在*Musurgia Universalis* **2**(1650)的10页和11页上列出五音符旋律的30个排列时，缺乏系统性使他陷入困境(参见习题5)：


(7)

但约翰·沃利斯懂得的多些。在*Discourse of Combinations*(1685)的117页上，如果令m<e<s，他以词典顺序正确地列出“messes”的60个颠倒字母顺序的词，而且在126页，他建议了字母顺序排列方法，他写道，“这样使我们可以更加确信，不会丢失任何东西”。

后边我们将看到，印度博学大师沙恩加第瓦和纳拉燕纳在13和14世纪已经建立起排列生成的一个理论，尽管他们的工作超越了他们的时代和留有模糊性。 53

关孝和表。关孝和(1642—1708)是一位有超凡能力的教师和研究者，他在日本17世纪对数学的研究进行了革命化。他研究从联立齐次方程中消去变量时，推导出了$a_1b_2-a_2b_1$和$a_1b_2c_3-a_1b_3c_2+a_2b_3c_1-a_2b_1c_3+a_3b_1c_2-a_3b_2c_1$的表达式，这是我们现在认识的行列式。1683年他出版了关于这个发现的小册子，介绍了以这种方式列出所有排列的巧妙方案，即它们中的一半是“活的”(偶)而另一半是“死的”(奇)。以$n=2$的情况开始，当“1 2”为活，“2 1”为死。对于$n>2$他列出以下规则：

1) 取$n-1$的每个活的排列，所有元素都增1，并把1插于前边。这规则产生了$\{1,\cdots,n\}$的$(n-1)!/2$个“基本排列”。

2) 由每个基本排列，通过转动和反射形成$2n$个其他排列：

$$a_1a_2\cdots a_{n-1}a_n,\ a_2\cdots a_{n-1}a_na_1,\cdots,\ a_na_1a_2\cdots a_{n-1} \tag{8}$$

$$a_na_{n-1}\cdots a_2a_1,\ a_1a_na_{n-1}\cdots a_2,\cdots,\ a_{n-1}\cdots a_2a_1a_n \tag{9}$$

如果n为奇数，第一行为活的，而第二行为死的。如果n为偶数，在每行中为交替地活，死，…，活，死。

例如，当$n=3$时，惟一的基本排列为123。因此123, 231, 312为活的，而321, 132, 213为死。因此，我们成功地生成一个3×3行列式的6项。$n=4$的基本排列是1234,1342,1432；而且，从比如1342开始，我们得到8项，即

$$+1342-3421+4213-2134+2431-1243+3124-4321 \tag{10}$$

交替地为活(+)和死(−)。因此，一个4×4的行列式包括项$a_1 b_3 c_4 d_2 - a_3 b_4 c_2 d_1 + \cdots - a_4 b_3 c_2 d_1$，以及16个其他项。

关孝和提出的对于排列生成的规则十分漂亮，但它有一个严重问题：当$n>4$时，它就无效了。他的错误似乎过了数百年仍未被认识。[参见三上义夫，*The Development of Mathematics in China and Japan* (1913)，191-199。关孝和全集(大阪：1974)，18-20，八五一～一四一，以及习题7-8。]

组合表。已知的、经历了时间洗涤而存在下来的、最早的穷尽的类似组合表，出现在写于公元600年前或更早的萨斯鲁塔著名的关于医药的梵语论文的最后一本书第63章。注意，药可以是甜、酸、咸、辣、苦和/或涩的。萨斯鲁塔的书巧妙地列出当这些量一次出现两个，三个，四个，五个，六个和一个时的情况(15, 20, 15, 6, 1, 6)。

巴斯卡拉在*Līlāvatī*的110～114节中重复了这个例子，并且发现，对于给定个数的长音节，同样的推理适用于6音节的诗步。但他简单地提到总数(6, 15, 20, 15, 6, 1)，而没有列出组合本身。在274和275节中，他发现数$(n)(n-1)\cdots(n-k+1)/(k(k-1)\cdots(1))$枚举组成(即有序的分划)以及组合；但再一次，他未给出表。

为避免罗嗦，以简单方式处理。

因为计算科学是一个无边的海洋。

54 ——巴斯卡拉(大约1150年)

一个孤立但有趣的类似组合表出现在由巴格达的阿尔-沙马瓦尔所写的著名代数课本 *Al-Bāhir fi'l-ḥisāb* (The Shining Book of Calculation)，当时他才19岁(1144)。在该书的末尾部分，他给出10个未知数的$\binom{10}{6}=210$个联立线性方程组的表：

阿尔–沙马瓦尔的阿拉伯原文

٦٥	٦٥٤٣٢١	ا
٧٠	٧٥٤٣٢١	ب
٧٥	٨٥٤٣٢١	جـ
	⋮	
٩١	١٠٩٨٧٦٤	زط
١٠٠	١٠٩٨٧٦٥	زي

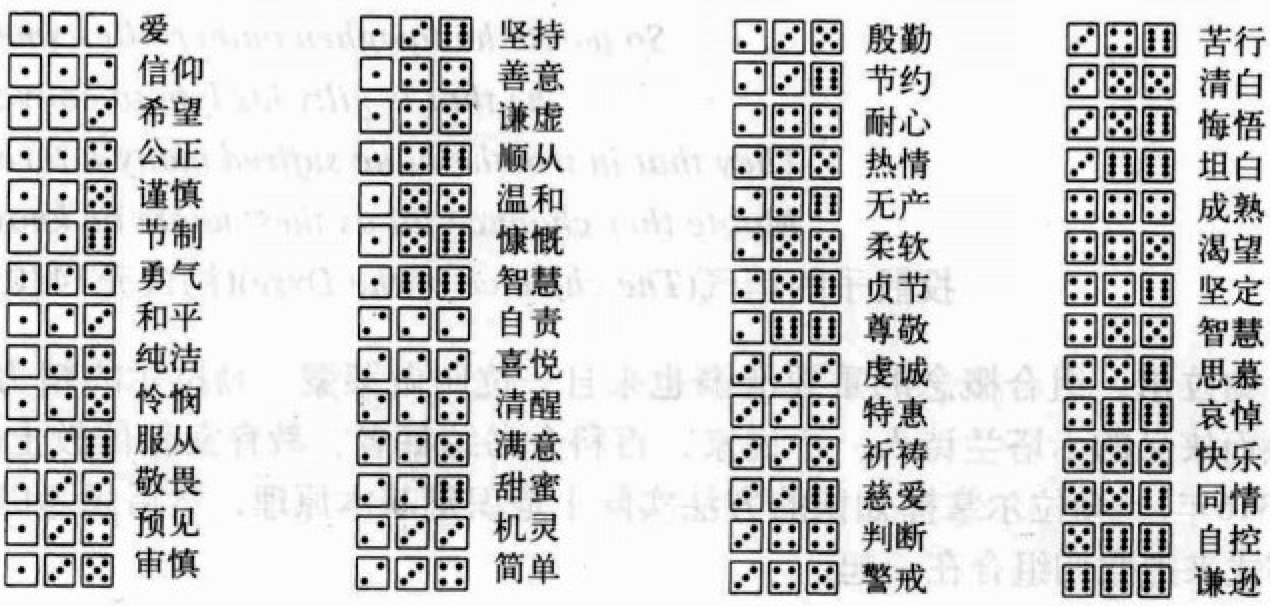
等价的现代记号

$$\begin{aligned}
&(1)\ x_1+x_2+x_3+x_4+x_5+x_6=65\\
&(2)\ x_1+x_2+x_3+x_4+x_5+x_7=70\\
&(3)\ x_1+x_2+x_3+x_4+x_5+x_8=75\\
&\qquad\vdots\\
&(209)\ x_4+x_6+x_7+x_8+x_9+x_{10}=91\\
&(210)\ x_5+x_6+x_7+x_8+x_9+x_{10}=100
\end{aligned} \tag{11}$$

其中每个方程代表在10个事物中一次取6个的每种组合。他的目的显然要证明超定方程组仍然可能有惟一的解—— 在此情况下，$(x_1, x_2, \cdots, x_{10})=(1, 4, 9, 16, 25, 10, 15, 20, 25, 5)$[沙拉·阿哈默德和罗什迪·拉塞德，*Al-Bāhir en Algèbre d' As- Samaw'al* (大马士革：1972)，77-82，٢٤٨-٢٣١]。

掷骰子。中世纪的欧洲也出现了少量的初等组合，突出的是同投掷三个骰子，

列出所有可能结果的问题相关。当然允许出现重复，从6个中选择3个，有$\binom{8}{3}=56$种方式。(当允许重复时，计算公式为$\binom{n+m-1}{m}$。——译者注)赌博被官方禁止，但这56种方式是比较有名的。大约公元965年，法国北方康布莱的主教维博尔德设计了一个叫做Ludus Clericalis的游戏，使得牧师们在保持虔诚的同时，做投骰子的游戏。他的想法是按照下表，把骰子的每个可能投掷结果与56种美德中的一种相关联：



游戏者轮流投掷，第一个人投掷获得各美德。在所有人投掷之后，获得最高美德的游戏者获胜。维博尔德说明，爱是所有美德中最好的。他给出一个复杂的计分系统，如果所有6个骰子的点数之和是21，可以把两个美德组合在一起，例如，可以以这种方式将爱+谦逊或者纯洁+智慧配对，这样的组合排在单个美德之上。他还考虑了游戏更复杂的变形，其中在骰子上用元音代替点，使得如果投掷出元音，则可以取得美德。 55

大约150年后由巴尔德里克在*Chronicon Cameracense*首先描述时，维尔博德的美德表是以词典顺序如上所示。[*Patrologia Latina* **134**(巴黎：1884), 1007-1016。]但是另一个中世纪的著作所介绍的骰子投掷可能性的排序方法却完全不同：



(12)

在这种情况下，作者知道如何处理重复的值，但有一个非常复杂的特别方法来处理所有骰子都不同的情况。[参见戴·理·贝尔豪斯，*International Statistical Review* **68**(2000), 123-136。]

有一首题为“Chaunce of the Dyse”的有趣的诗，传说由约翰·里德盖特

(John Lydgate)写于15世纪初期，用于各种联欢会。它的开头诗句邀请每人掷三个骰子，而后的其余语句以递减词典顺序对从⚅⚅⚅，到⚅⚅⚄，到…，到⚀⚀⚀的组合进行索引，给出轻松地描述投骰者的56个特征框架。[完整的文字由(埃·珀·汉默蒙德发表于*Englische Studien* **59**(1925)，1-16；翻译成现代英语形式将是合意的。]

I pray to god that euery wight may caste
Vpon three dyse ryght as is in hys herte
Whether he be rechelesse or stedfaste
So moote he lawghen outher elles smerte
he that is gilty his lyfe to converte
They that in trouthe haue suffred many a throwe
Moote ther chaunce fal as they moote be knowe.

投骰子的运气(*The chaunce of the Dyse*)(约公元1410年)

莱蒙·勒拉尔。组合概念的重要涟漪也来自一位叫做莱蒙·勒拉尔的精力充沛的和狂热而侠义的卡塔兰诗人、小说家、百科全书编纂者、教育家和传教士(约1232年—1316年)。勒拉尔掌握知识的方法实际上是识别基本原理，然后思考以所有可能的方式来把它们组合在一起。

例如，在*Ars Compendiosa Inveniendi Veritatem*(约1274年)的一章中，开始枚举上帝的16种属性：善良、伟大、永恒、权力、智慧、爱、美德、正确、光荣、完善、公正、慷慨、怜悯、谦卑、主权、以及耐心。然后，勒拉尔为每个写出大约80字
56 的$\binom{16}{2}=120$篇短文，并认为上帝的善良同伟大有关，上帝的善良同永恒有关，等等；并以上帝的主权同耐心有关结束。在另一章，他考虑7种美德(信仰、希望、善行、公正、谨慎、坚韧、自制)和7种罪恶(暴食、色欲、贪婪、懒惰、骄傲、嫉妒、发怒)，并用$\binom{14}{2}=91$分章依次成对地讨论。其他章以类似的方式系统地分成为$\binom{8}{2}=28$, $\binom{15}{2}=105$, $\binom{4}{2}=6$, 以及$\binom{16}{2}=120$个小节。(人们不知道，如果他熟悉维尔博德的56种美德表，会发生什么。他将产生出对于所有$\binom{56}{2}=1540$对的评述吗？)

勒拉尔通过画出图44所示循环的框图来说明他的方法。在这张图中左边的圆圈表示神仙，命名了16个神授的属性——实际上和早先列出的相同，除了现在把爱(不道德amor)称为意愿(情欲)外，另外最后四个现在分别叫做简单性，等级，怜悯和主权。每个属性赋予一个代码字母，而这张图把它们的相互关系描述为在顶点(B, C, D, E, F, G, H, I, K, L, M, N, O, P, Q, R)上的完全图K_{16}。右边的图表示美德和罪恶，显示出7种美德(b, c, d, e, f, g, h)和7种罪恶(i, k, l, m, n, o, p)的交叉。在原

稿中，美德以蓝色出现而罪恶以红墨水给出。注意，在此情况下，他的图描述了两个独立的完全图K_7，一种颜色对应一个完全图(他不再麻烦地对每个美德和每个罪恶进行比较，因为每种美德显然比每种罪恶好。)

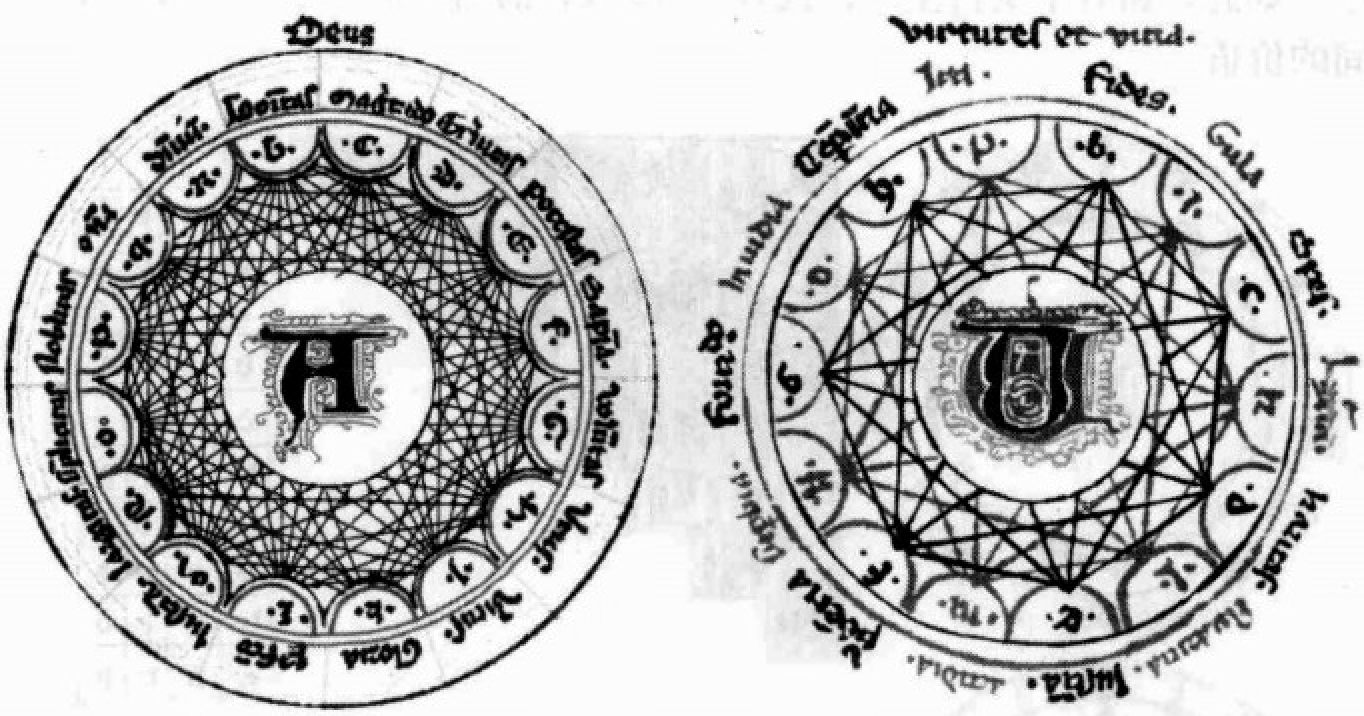


图44 由莱蒙·勒拉尔给出的一部手稿中对于1280年时的地方行政官的说明
[取自他的Ars Demonstrativa，Biblioteca Marciana，VI 200, folio 3^v。]

勒拉尔使用相同的方法来写药品：代替连接神学的概念，他的*Liber Principiorum Medicinæ*(大约1275年)考虑了病症和处治的组合。他还写了关于哲学、
逻辑、法学、天文学、动物学、几何学、修辞学以及骑士制度的图书，总数在200 57
种以上。然而，必须承认，他的许多内容都是重复的；现代数据压缩技术大概将大大压缩勒拉尔的结果。比如说，压缩到比亚里士多德的成果小得多的规模。

最终他决定集中主要工作在九种事物组中，来简化他的系统。比如说，看图45，那里圆圈A现在仅列出前九种上帝的属性(B, C, D, E, F, G, H, I, K)。$\binom{9}{2}=36$种关联对(BC, BD, …, IK)出现在这个圆圈右边的阶梯形图表中。通过加上另外两个属性，即耐心和同情心——以及另外两个罪恶，即撒谎和自相矛盾——他能够用同一张图表来处理美德同美德、以及罪恶和罪恶。他还提出，使用同一图表在有九个候选人的选举中进行投票的一个有趣方案。[参见伊·麦克里安和约·伦敦，*Studia Lulliana* **32**(1992)，21-37。]

在图45左下方圆圈内的三角形表示勒拉尔方法的另一个关键方面。三角形(B，C，D)代表(差别，和谐，对立)，三角形(E，F，G)代表(开始，中间，结束)，以及三角形(H，I，K)代表(大于，等于，小于)。这三种相互交叉的K_3的出现代表三种类型的3值逻辑。勒拉尔很早时就实验了其他这样的三元组，主要为(真，未知，假)。通过考虑他是如何处理四个基本元素(地球，空气，火，水)的组合，我们就能对他如何使用三角形有一个了解：所有四种元素都不同，地球与火是相和谐的，它同空气和谐，同水和谐，同地球和谐；而地球同空气是相对立的，火与水是相对立的。这些考虑就完成了关于三角形(B，C，D)的分析。再看三角形(E，F，G)，

他说明自然界的各种进程都以支配另一个元素的一个元素开始；然后出现一个过渡或中间状态，直到达到一个目标为止，像空气变暖这样。对于三角形(H，I，K)，他认为，一般相对于它们的“范围”、“速度”和“崇高性”，有火>空气>水>地球；尽管如此，相对于支持生命，也有空气>火，而当它们一起工作时，空气与火有相同的价值。

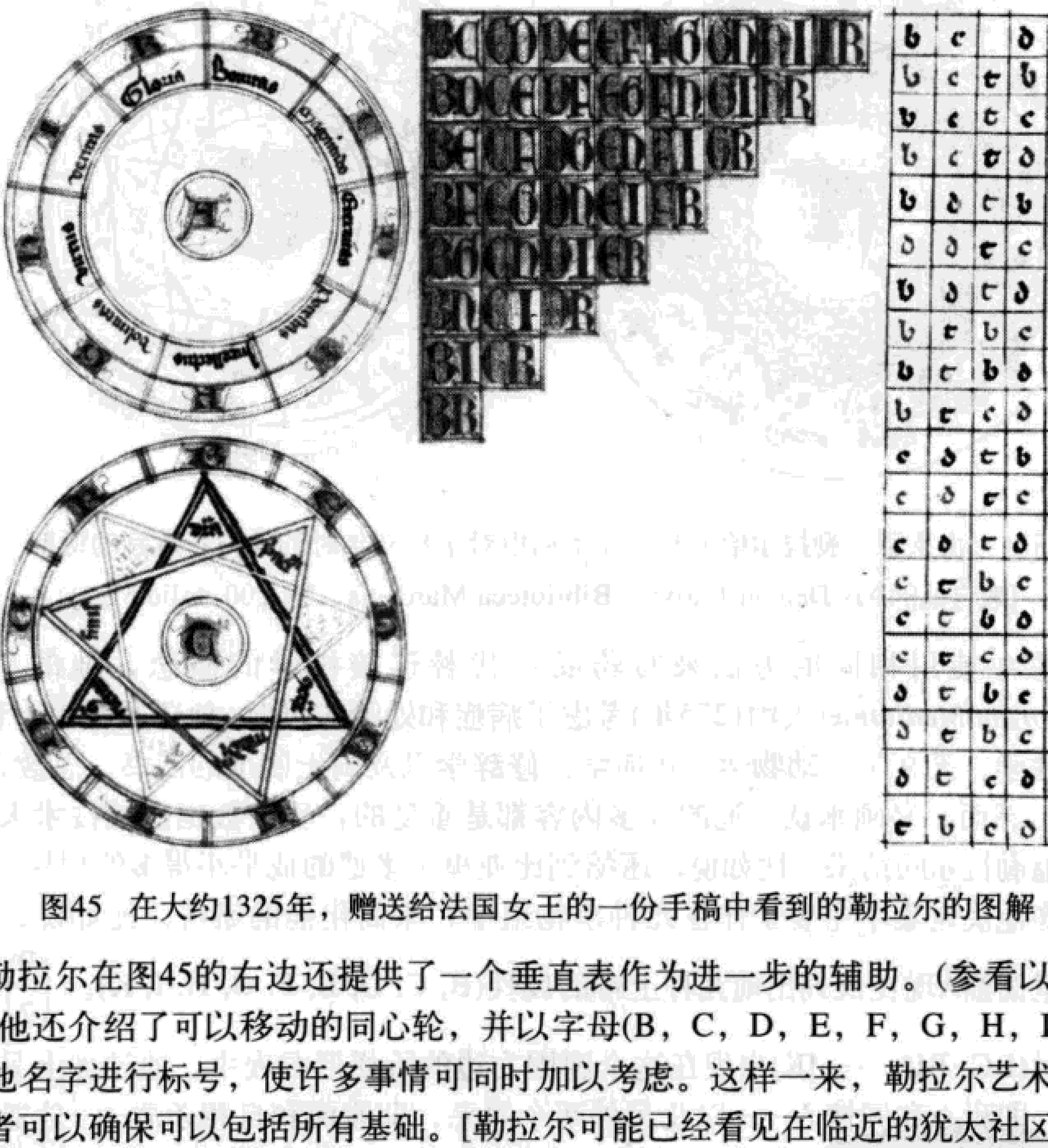


图45 在大约1325年，赠送给法国女王的一份手稿中看到的勒拉尔的图解

勒拉尔在图45的右边还提供了一个垂直表作为进一步的辅助。(参看以下习题11)。他还介绍了可以移动的同心轮，并以字母(B，C，D，E，F，G，H，I，K)以及其他名字进行标号，使许多事情可同时加以考虑。这样一来，勒拉尔艺术的忠实实践者可以确保可以包括所有基础。[勒拉尔可能已经看见在临近的犹太社区中使用了类似的轮子；参见莫·埃德尔 *J. Warburg and Courtauld Institutes* **51** (1988)，170-174和底片16-17]。

好多个世纪之后，阿塔纳西尤斯·克彻尔发表了勒拉尔系统的一个扩充作为标题为*Ars Magna Sciendi sive Combinatoria*的大部头的一部分，(阿姆斯特丹：1669)，并有五个可移动的轮子伴随着该书的第173页。克彻尔还通过提供完全的二部分图$K_{m,n}$的图示，来扩充勒拉尔的完全图K_n的全部图。例如，图46取自克彻尔的书第171页，而第170页中包含了$K_{18,18}$的壮丽的图。

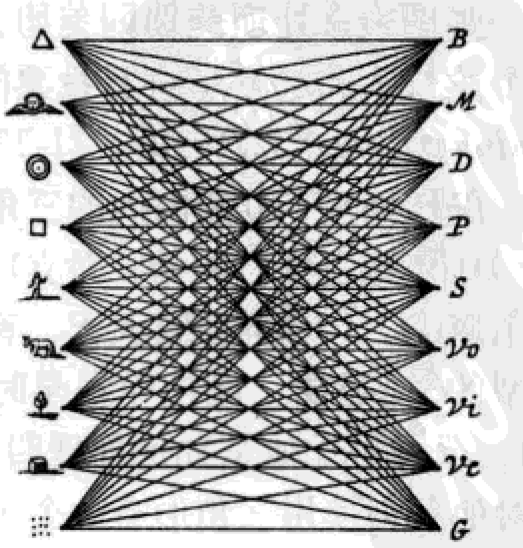


图46 由阿塔纳西尤斯·克彻尔于1669年给出的$K_{9,9}$

它是值得研究和有创造力的艺术。
当以所有可能的方式把思想组合在一起时，
新的组合开始沿着新的通道思维，
引导人们去发现新鲜的真理和论断。
马丁·加德纳，*Logic Machines and Diagrams*(1958)

类勒拉尔方法的最广泛的现代发展或许是由约瑟夫·斯契尔令格所著的*The Schillinger system of Musical Composition*(纽约：Carl Fischer 1946)。这是一本从组合方面介绍韵律，旋律，和谐，配合旋律，作曲，交响乐等等的理论的著名两卷集。例如，在第56页，斯契尔令格以平易改动的格雷码顺序列出$\{a, b, c, d\}$的24个排列(算法7. 2. 1. 2P)；然后在57页他应用它们不是来定调的高低而是来定韵律，确定音符的连续期限。在364页，他显示了对称的环 58~59

$$(2, 0, 3, 4, 2, 5, 6, 4, 0, 1, 6, 2, 3, 1, 4, 5, 3, 6, 0, 5, 1) \tag{13}$$

这是对于7个对象$\{0, 1, 2, 3, 4, 5, 6\}$的2组合的一个通用环。换言之，(13)是K_7的一个欧拉迹；所有$\binom{7}{2}=21$对数字恰好出现一次。这种模式是作曲家的产物。但是我们要感谢斯契尔令格更好的学生们(像乔治·吉尔斯文，不使自己完全沉溺于美学的严格数学意义上。

塔克奎特，范·斯库廷和伊泽奇奎尔多。17世纪50年代出版了和我们故事有关的另外三本书。安德烈·塔克奎特写了一本流行的教材*Arithmeticæ Theoria et Praxis*(Louvain：1656)。这本书在后来的50年中多次重印和修订，在接近结尾处的376和377页，他给出了一个步骤，用它来列举出一次两个、一次三个等组合的清单。

弗朗斯·范·斯库廷的*Exercitationes Mathematicæ*(Leiden：1657)更先进。在373页，他以一个有趣的布局列出了所有组合

$$\begin{array}{c} a \\ \hline b.\ ab \\ \hline c.\ ac.\ bc.\ abc \\ \hline d.\ ad.\ bd.\ abd.\ cd.\ acd.\ bcd.\ abcd \end{array} \tag{14}$$

而且，在下几页他继续把他的模式扩充到字母e, f, g, h, i, k，“乃至无穷”。在376页，他发现能够以(2, 3, 5, 7)来代替(14)中的(a, b, c, d)，得到超过1的210的因子：

$$\begin{array}{c} 2 \\ \hline 3\ \ 6 \\ \hline 5\ \ 10\ \ 15\ \ 30 \\ \hline 7\ \ 14\ \ 21\ \ 42\ \ 35\ \ 70\ \ 105\ \ 210 \end{array} \tag{15}$$

在下一页，他把这个思想推广到

$$\begin{array}{c} a \\ \hline a.\ aa \\ \hline b.\ ab.\ aab \\ \hline c.\ ac.\ aac.\ bc.\ abc.\ aabc \end{array} \tag{16}$$

因此，允许有两个a。尽管他并不真正理解这个扩充，他的下一个例子

$$\begin{array}{c} a \\ \hline a.\ aa \\ \hline a.\ aaa \\ \hline b.\ ab.\ aab.\ aaab \\ \hline b.\ bb.\ abb.\ aabb.\ aaabb \end{array} \tag{17}$$

60 是拙劣的，表示在那时他的知识的局限。(参见习题13。)

在第411页，范·斯契尔令格发现，可以把权$(a, b, c, d)=(1, 2, 4, 8)$赋予(14)中，在相加之后得到

$$\begin{array}{c} 1 \\ \hline 2\ \ 3 \\ \hline 4\ \ 5\ \ 6\ \ 7 \\ \hline 8\ \ 9\ \ 10\ \ 11\ \ 12\ \ 13\ \ 14\ \ 15 \end{array} \tag{18}$$

但他没有看出同二进制算术的联系。

施巴斯蒂安·伊泽奎尔多的两卷著作*Pharus Scientiarum*(里昂：1659)，即“科学的灯塔”，包括了标题为关于组合学论断29(Disputatio 29. *De Combinatione*)的组织得很好的讨论。对于斯坦利的十二折方法的四个关键部分，即n元组，n个变异，n多重组合，以及出现在表7.2.1.4-1的头两行和头两列的m个对象的n组合，他都给出了详细的讨论。

在*De Combinatione*的81~84节中，对于$2\leqslant n\leqslant 5$和$n\leqslant m\leqslant 9$，他总是以词典顺序列出对m个字母一次取n个的所有组合；而且对于$m=10$和20，在$n=2$和3的情况下，他也对它们造表。但当他列出m个事物中一次取n的$m^{\underline{n}}$个变异时，他选择了一个更复杂的顺序(参见习题14)。

伊泽奎尔多最先发现对于不限制重复的m个事物一次取n个的组合公式$\binom{m+n-1}{n}$，这个规则出现在他的书§48-§51中。但在§105中，当他试图$n=3$的情况下列出所有这样的组合时，他却不知道有这样做的一个简单方法。事实上，对于$m=6$，他列出的56种情况像是(12)的旧的、笨拙的顺序。

对于允许重复的组合在1713年詹姆斯·伯努利的*Ars Conjectandi*“猜测的艺术”出版之前，没有很好地理解。在第2部分第5章中，伯努利以词典顺序简单地列出可能性，并且以归纳法作为一个容易的推论，证明了公式$\binom{m+n-1}{n}$。[尼科罗·塔尔塔格里亚偶然在*General trattato di numeri, et misure* **2** (威尼斯：1556)，17^{r}和69^{v}中，也已经接近于发现这个公式。同样，在13世纪的著作*Fiqh al-Ḥisāb.*中，马格里布数学家伊本·姆恩伊姆也发现了]。

零的情况。在结束关于组合学早期工作的讨论之前，我们不应忘记由约翰·沃利斯在*Discourse of Combinations*(1685年)第110页所采取的一个很小的但很杰出的步骤。他特别地考虑了在m个事物中一次取0个的组合：“它是明显的，即，如果我们将取无，即，如果我们留下所有的；那就只能有一种情况。这就是暴露的事物的数量”。而且，在113页，他知道$\binom{0}{0}=1$。因为取所有事物，或者留下所有事物，是一个相同的情况。”

然而，当他给出对于$n\leqslant 24$的$n!$的表时，他却没有得到$0!=1$的结论，或者对于空的集合恰好有一种排列。

纳拉燕纳的工作。感谢帕尔玛南德·辛格的英文翻译[*Gaṇita Bhāratī* **20**(1998)，25-82；**21**(1999)，10-73；**22**(2000)，19-85；**23**(2001)，18-82； 61
24(2002)，35-98]，近来才使印度以外的学者们第一次详细知道由纳拉燕纳·潘迪塔于1356年写就的以<演算的安逸欢乐>(*Gaṇita Kaumudī*)为题的著名专著；也可参看楠叶隆德的博士论文，布朗大学(1993)。纳拉燕纳的著作第13章小标题为*Aṅka Pāśa*（"数的连接"）专门讨论组合生成。确实，尽管这一章的97个"箴言"是相当神秘，但它们比世界其他地区提前几百年介绍了关于这个课题的广泛理论发展。

例如，纳拉燕纳在箴言49-55a中处理了排列的生成，他给出了递减协词典顺序列出集合的所有排列的一些算法，连同对于一个给定的排列排阶和对一个给定的序列数去阶的一些算法。但这些算法，早在一个多世纪以前，由沙恩加第瓦在著名论著"音乐的钻石矿"（*Saṅgītaratnākara*）§1. 4. 60-71中给出，沙恩加第瓦因此实际上已经发现了正整数的阶乘表示。纳拉燕纳继续在箴言57-60中推广沙恩加第瓦的算法，使一般的多重集合也能容易地排列，例如，他列出{1，1，2，4}的排列为

1124, 1214, 2114, 1142, 1412, 4112, 1241, 2141, 1421, 4121, 2411, 4211

还是按递减的协词典顺序。

纳拉燕纳的箴言88-92讨论了组合的系统生成。除了说明{1, ⋯, 8}一次取3个的组合，即

{678, 578, 478, ⋯, 134, 124, 123}

之外，他也考虑了以相反的顺序(递增协词典顺序)表示这些组合的二进位串，并且推广了巴特托特帕拉十世纪的方法：

(11100000, 11010000, 10110000, ⋯, 00010011, 00001011, 00000111)

他几乎但未十分完全地发现定理7. 2. 1. 3L。

可排列的诗。现在让我们转到吸引了17世纪许多杰出数学家注意的奇怪问题，因为它显露出那时在欧洲组合数学知识的状态。一位名叫贝尔纳德·鲍休斯的犹太牧师以拉丁六韵步作了一行著名的诗献给圣母玛丽亚：

Tot tibi sunt dotes, Virgo, quot sidera cælo (19)

["哦圣母啊，你的美德好比天上的繁星一般，数也数不清"。参见他的*Epigrammatum Libri* V(戈罗格尼：1615)，49。]他的诗启示了Lourain大学的教授埃里休斯·普提亚奴斯来写一本题目为*Pietatis Thaumata* (Antwerp：1617)的书，介绍了鲍休斯著作中的1022个排列。例如，普提亚奴斯写道：

107　Tot dotes tibi, quot cælo sunt sidera, Virgo.
270　Dotes tot, cælo sunt sidera quot, tibi Virgo.
329　Dotes, cælo sunt quot sidera, Virgo tibi tot.
384　Sidera quot cælo, tot sunt Virgo tibi dotes.　　(20)
725　Quot cælo sunt sidera, tot Virgo tibi dotes.
949　Sunt dotes Virgo, quot sidera, tot tibi cælo.
62　1022　Sunt cælo tot Virgo tibi, quot sidera, dotes.

他于1022处停下来，因为1022是在普托列米著名的天空目录中可见的星星的个数。

以这样一种方式对词进行排列的思想在那时是非常有名的。这样一种词游戏是尤里乌斯·斯卡里格在*Poetices Libri Septem*(里昂：1561)，第2卷第30章中所谓的“变幻无常诗行”。拉丁语言本身倾向于像(20)这样的排列，因为拉丁词的结尾一般都定义每个名词的功能，使词的相对顺序对于一个句子的意义来说不像它在英语中那样重要。然而普提亚奴斯指出，他已经特别地避免不适当的排列，例如

Sidera tot cælo, Virgo, quot, sunt tibi dotes,　　(21)

因为它们将对圣女的美德设置上限，而不是下限。[参见他的书12页和103页。]

当然，有8!=40 320种方式对(19)的词进行排列。但问题不在这里；大多数方式不“顿挫合拍”。普提亚奴斯的1022首诗的每一个都严格遵守经典的六韵步的规则。自荷马和弗吉尔的时代起，希腊和拉丁的诗人们就遵从这些规则，即：

i) 每个词由长(—)或短(⌣)音节组成。

ii) 每行的音节属于32个模式之一。

$$\left\{\begin{matrix}-\smile\smile\\--\end{matrix}\right\}\left\{\begin{matrix}-\smile\smile\\--\end{matrix}\right\}\left\{\begin{matrix}-\smile\smile\\--\end{matrix}\right\}\left\{\begin{matrix}-\smile\smile\\--\end{matrix}\right\}-\smile\smile\left\{\begin{matrix}-\smile\\--\end{matrix}\right\}. \tag{22}$$

换言之，有6个音步呎，其中前四个中的每一个是在(5)的术语下的扬抑抑格，或者是扬扬格；第五个脚应是扬抑格，而最后一个应该是扬抑格或者是扬扬格。

一般说来，在拉丁语诗歌中长和短音节的规则是稍微复杂的，但鲍休斯的诗歌的八个词可以由下列模式来表征：

$$\begin{gathered}\text{tot} = -,\ \text{tibi} = \left\{\begin{matrix}\smile\smile\\\smile-\end{matrix}\right\},\ \text{sunt} = -,\ \text{dotes} = --,\\ \text{Virgo} = \left\{\begin{matrix}-\smile\\--\end{matrix}\right\},\ \text{quot} = -,\ \text{sidera} = -\smile\smile,\ \text{cælo} = --.\end{gathered} \tag{23}$$

注意，当使用词“tibi”或“Virgo”时，诗人们有两种选择。因此，例如(19)适合于六韵步的模式：

$$\overset{-\ \smile\ \smile}{\text{Tot ti-bi}}\quad \overset{-\ \ -}{\text{sunt do-}}\quad \overset{-\ \ -}{\text{tes, Vir-}}\quad \overset{-\ \ -}{\text{go, quot}}\quad \overset{-\ \smile\ \smile}{\text{si-de-ra}}\quad \overset{-\ -}{\text{cælo.}} \tag{24}$$

(扬抑抑，扬扬，扬扬，扬扬，扬抑抑，扬扬：“dum-diddy dum-dum dum-dum dum-dum dum-diddy dum-dum”。当把这些词读出时，逗号代表稍微停顿，称为“停顿”；这里我们不关心它们，尽管普提亚奴斯小心地把它们插入到他的1022个排列的每一个当中)。

现在自然地提出一个问题：如果我们随机排列鲍休斯的词，它们符合的格律

的差异是什么？换言之，给定在(23)中的音节模式，有多少排列服从规则(i)和(ii)？戈·威·莱布尼茨在*Dissertatio de Arte Combinatoria*(1666年)中提出其他问题的同时，提出了这个问题。这一著作是莱布尼茨在莱比锡大学申请职务时发表的。那时他才19岁，主要靠自学，对于组合学的理解是十分有限的。例如，他相信{ut, ut, re, mi, fa, sol}有600种排列，而{ut, ut, re, re, mi, fa}有480种排列。他甚至提出，(22)代表76种可能性，而不是32种。[参见在他的问题6中的§5和§8]。 63

但是莱布尼茨确实认识到，在许多排列“无用”时，计算“有用”的所有排列的个数的通用方法是值得的。他考虑了普提亚奴斯诗歌的好多例子，正确地枚举了某些较简单的例子，但当词较复杂时，产生许多错误。尽管他提到普提亚奴斯的工作，但他不打算枚举(19)的符合格律的排列。

一些年之后，由简·普勒斯特在*Élémens des Mathématiques* (巴黎：1675)一书的342-438页介绍了更为成功得多的方法。普勒斯特给出了一个清晰的剖析，即恰有鲍休斯的诗文的2196个排列将产生一个适当的六韵步。然而，他不久就认识到，他忘记了统计相当多的情况——包括在(20)中编号为270,384和725的那些。所以，当他于1689年发表*Nouveaux Élémens des Mathématiques*时，他完全重写这部分内容。普勒斯特的新书127-133页专门致力于证明，可符合格律的排列的真正个数为3276，比他以前给出的的总和几乎增加了50%。

与此同时，约翰·沃利斯在他的*Discourse of Combinations* (伦敦：1685)的118-119页，讨论了这个问题，发表出来作为他的*Treatise of Algebra*的补充。在说明了为什么他相信正确的数字为3096后，沃利斯承认，他可能忽略了某些可能性和/或对某些情况统计一次以上。“但是现在，我不能辨别出究竟哪一个是正确的”。

对沃利斯的工作，一个佚名评审员指出，在音步上正确排列的真正个数实际上是2580——但他没有给出证明。[*Acta Eruditorum*, **5**(1686), 289]。这个评审员几乎肯定是戈·威·莱布尼茨本人，尽管在莱布尼茨大量未发表的手稿中找不到关于数2580后边的推理的提示。

最后，詹姆斯·伯努利进入了这个图画。在他1682年作为巴塞尔大学哲学院院长的就职讲演中，他提到了托梯比枚举问题并且指出，为得到一个正确的答案，仔细的分析是必要的。他说，它是3312(!)。他的证明出现在他死后才出版的*Ars Conjectandi*, (1713)第一版的79-81页。[顺便说，那些页从他那著名的书稍后的版本中以及从他的全集中被删掉了，因为实际上他并不打算发表它们。审稿者错误地把它们插进来了。参见*Die Werke von Jakob Bernoulli* **3**(巴塞尔：Birkhauser, 1975), 78,98-106,108,154-155。]

那么，谁对呢？有2106种符合格律的排列呢？还是3276，或3096，或2580，还是3312？威·阿·惠特沃思和威·厄·哈特利在*The Mathematical Gazette* **2**(1902), 227-228中重新考虑了这个问题，其中他们每个人都给出了漂亮的论证，并且得出结论，上面那些数没有一个是真正的答案。他们共同的答案是2880，这代表了头一次，有两名数学家对于这个问题独立地得出相同的结论。 64

但以下的习题21和22揭示了真理：伯努利被证明是正确的，而其他人都是错的。而且对伯努利系统和仔细的内嵌式的3页推导的研究表明，他的成功主要是因为他

忠实地坚持我们现在叫做*回溯的方法*。在7.2.2节中我们将彻底地研究回溯方法，我们也将看到，作为*精确覆盖问题*的一个特殊情况，tot-tibi问题可容易地求解。

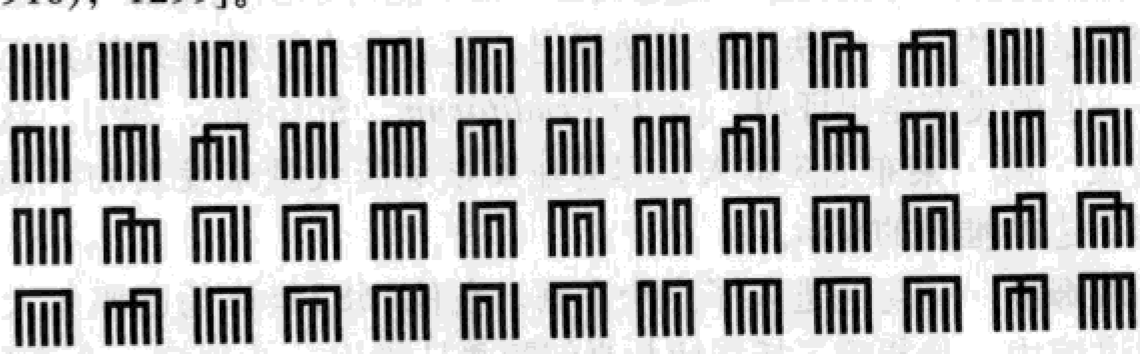
> 即使是最聪明的和最审慎的人经常也会由于逻辑学家所说的对情况的不充分的枚举而吃苦头。
>
> ——詹姆斯·伯努利(1692年)

集合分划。集合的分划似乎首先是日本研究的，在大约公元1500年时叫做〈源氏–香〉的室内游戏在上层人中十分流行。收集香的主人秘密地选择5包香，其中一些可能是相同的，他一次烧它们当中的一包。客人们将尝试识别哪些香相同，而哪些香不同。换言之，他们将猜测{1，2，3，4，5}的ϖ_5=52个分划中哪一个被主人选中。



图47 用来代表在16世纪的日本中的集合分划的图[来自琦玉大学的矢野环的选集]

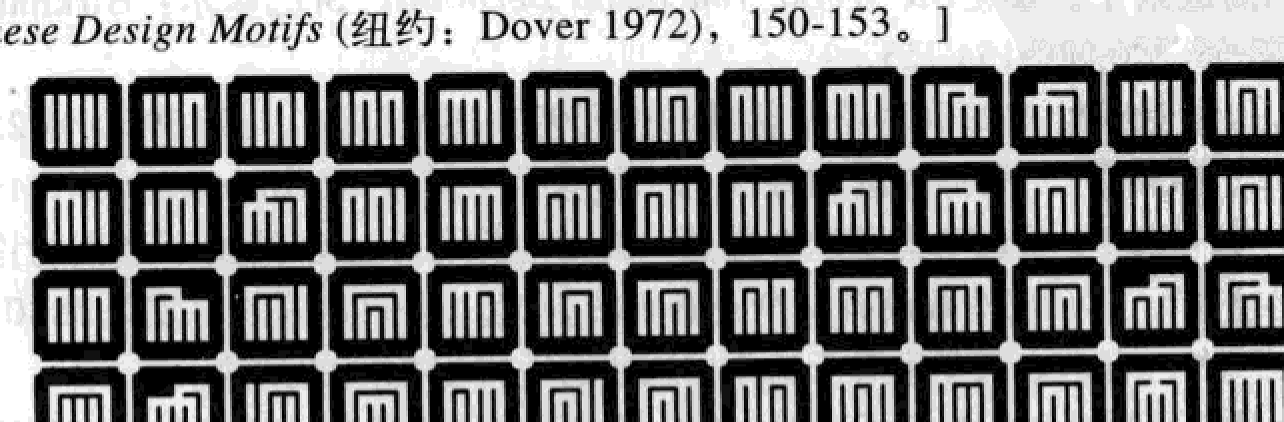
不久之后，人们习惯于以类似于图47所示框图来代表52种可能的结果。例如，当从右向左读时，该图的最上方框图将表示头两个香是相同的，因而后三个也相同。这样，分划为12|345。类似地，其他两个框图表示分划124|35和1|24|35的图解方式。为辅助记忆，在紫式部夫人著名的11世纪<源氏-香>(*Tale of Genji*)的一章之后，对于52种模式的每一个按照下列顺序赋予名字*Encyclopedia Japonicae* (东京：三省堂，1910)，1299]。



(25)

(再次，如我们在许多其他例子中所见到的，这些可能性并未以任何特定的逻辑顺
65 序来安排。)

这些源氏香模式的有趣本质导致许多家庭把它们作为预言的顶峰。例如，(25)的下列风格化变形可在20世纪初和服模式的标准分类中找到。[参见安达父江，*Japanese Design Motifs* (纽约：Dover 1972)，150-153。]



18世纪初期，关孝和和他的学生们受到对于ϖ_5=52已知结果的激励，开始研究

对于任意的n，集合分划ϖ_n的个数。松永良弼发现，对于$1\leqslant j\leqslant t$，当具有大小为n_j的k_j个子集，且$k_1 n_1+\cdots+k_t n_t=n$时，集合分划个数的公式。(参见习题1.2.5-21的答案)。他还发现了基本的递归关系7.2.1.5-(14)。即

$$\varpi_{n+1}=\binom{n}{0}\varpi_n+\binom{n}{1}\varpi_{n-1}+\binom{n}{2}\varpi_{n-2}+\cdots+\binom{n}{n}\varpi_0 \qquad (26)$$

由这个公式，可以容易地计算ϖ_n的值。

松永良弼的发现在有马赖僮的书*Shūki Sanpō*于1769年出版之前一直未发表。该书的问题56要求读者对于n，求解方程“ϖ_n=678 570”。而有马赖僮的答案，详细地给出细节(他并且归功于松永良弼)，是n=11。

不久之后，坂正永在他的著作*Sanpō-Gakkai*(1782)研究了把n集合分划成为k个子集的方式数$\left\{{n \atop k}\right\}$。他发现了递归公式

$$\left\{{n+1 \atop k}\right\}=k\left\{{n \atop k}\right\}+\left\{{n \atop k-1}\right\} \qquad (27)$$

而且对于$n\leqslant 11$，把结果造成表。詹姆斯·斯特林在他的*Methodus Differentialis*(1730)中，在纯粹代数的范畴内发现了数$\left\{{n \atop k}\right\}$，因此坂正永是第一位认识它们的组合意义的人。

随后由本田利明设计了列出集合分划的一个有趣的算法(参见习题24)，关于源氏–香进一步的细节以及它同数学史的关系可以在日文的矢野环的文章*Sugaku Seminar* **34**，11(Nov. 1995)，58-61；**34**，12(Dec. 1995)，56-60中找到。

除了三个孤立的偶然事件外，在欧洲直到很晚，集合分划实际上都尚未为人所知。首先是乔治(George)和/或理查德·普廷哈姆(Richard Puttenham)于1589年发 66
表了*The Arte of English Poesie*，在该书的70-72页包括了类似于源氏–香的那些框图。例如，七个框图


(28)

用来说明5行诗可能的音韵方案。“其中的一些它们要比其他的听起来粗糙和不大舒服”。但这个看起来有趣的表是不完整的(参见习题25)。

其次，戈·威·莱布尼茨于17世纪末期的一个未发表的手稿表明，他已经尝试计算把$\{1,\cdots,n\}$分划成为3个或4个子集的方式数，但几乎是不成功的。他以一个非常笨拙的方法枚举$\left\{{n \atop 2}\right\}$，但并没有使他容易地看到$\left\{{n \atop 2}\right\}=2^{n-1}-1$。仅对于$n\leqslant 5$，他尝试了计算$\left\{{n \atop 3}\right\}$和$\left\{{n \atop 4}\right\}$。但产生了好多个数值的遗漏导致不正确的答案。[参见伊·克·诺布罗赫，*Studia Leibnitiana Supplementa* **11**(1973)，229-233；**16**(1976)，316-321。]

集合分划在欧洲的第三个出现具有完全不同的性质。约翰·沃利斯的

Discourse of Combinations(1685)的第三章专门讨论“整除部分”，即数的真因子问题，他特别研究把给定的整数因式分解成所有方式的集合。这个问题等价于对于多重分划的研究。例如，当p，q和r是素数时，p^3q^2r的因式分解实质上和$\{p, p, p, q, q, r\}$的分划是相同的。沃利斯设计了列出给定整数n的所有因式分解的卓越算法，它实质上是预先处理算法7. 2. 1. 5M(参见习题28)。但他没有研究当n是一个素数的幂(等价于整数分划)或当n是非平方(等价于集合分划)的重要特殊情况。因此，尽管沃利斯能够解决更一般的问题，但是它的复杂性却使他未能发现分划数，贝尔数，或斯特林子集数，或者设计出生成整数分划或集合分划的简单算法。

整数分划。整数分划出现在人们的视线里甚至更慢些。维博尔德主教(约在965年)知道把n分划成恰好三个≤6的部分。伽利略也是，他对于这些分划写了一份备忘录(约在1627年)，而且研究了三个骰子的投掷出现频率。[“Sopra le scoperte de i dadi”在伽利略的*Opere*, Volume 8，591-594中，他以递减词典顺序列出分划]。

梅森在*Traitez de la Voix et des Chants* (1636)的130页列出把9划分成任意部分的分划。对于每个分划$9=a_1+\cdots+a_k$，他也计算了多项式系数$9!/(a_1!\cdots a_k!)$；如我们已经看到的，他对计算各种旋律感兴趣，而且他知道，9个音符$\{a, a, a, b, b, b, c, c, c\}$有$9!/(3!3!3!)=1680$个旋律。但他却未提到8+1和3+2+1+1+1+1的情况，大概是因为他没有以任何系统的方式列出这些可能性。

莱布尼茨在*Dissertatio de Arte Combinatoria* (1666)的问题3中考虑了两部分分
67 划，他未发表的笔记表明他随后花费了相当多时间试图枚举有三个或三个以上求和项的分划。他把它们叫做“断片”或者(不太经常)“扯裂”——当然在拉丁语中——有时叫做“部分”或“分散部”甚至“分划”。他对它们感兴趣主要是因为它们同单项式对称函数$\sum x_{i_1}^{a_1}x_{i_2}^{a_2}\cdots$的联系。但除了在三个求和项的情况之外，当他几乎(但还不完全)发现习题7. 2. 1. 4-31中$\left|{n \atop 3}\right|$的公式时，他的许多企图几乎全盘失败。例如，他粗心地只统计出8的21个分划，而忘记了2+2+2+1+1，而对于$p(9)$，在丢掉了3+2+2+2,3+2+2+1+1，2+2+2+1+1+1,2+2+1+1+1+1+1之后，他只得到26个——尽管事实上他试图以递减的词典顺序系统地列出分划。[参见埃·克诺布罗赫，*Studia Leibnitiana Supplementa* **11**(1973)，91-258；**16**(1976)，255-337；*Historia Mathematica* **1** (1974))，409-430。]

阿布拉汉姆·德·莫伊弗勒在他的论文“求一个无穷多项式任何给定次幂，或者求同样多项式的给定根的一种方法”[*Philosophical Transactions* **19**(1697)，619-625和图5]中，获得了对于分划的第一次真正的成功。他证明了$(az+bz^2+cz^3+\cdots)^m$的z^{m+n}的系数对于n的每个分划都有一项；例如，z^{m+6}的系数是

$$\binom{m}{6}a^{m-6}b^6+5\binom{m}{5}a^{m-5}b^4c+4\binom{m}{4}a^{m-4}b^3d+6\binom{m}{4}a^{m-4}b^2c^2$$
$$+3\binom{m}{3}a^{m-3}b^2e+6\binom{m}{3}a^{m-3}bcd+2\binom{m}{2}a^{m-2}bf+\binom{m}{3}a^{m-3}c^3 \qquad (29)$$
$$+2\binom{m}{2}a^{m-2}ce+\binom{m}{2}a^{m-2}d^2+\binom{m}{1}a^{m-1}g$$

如果我们置a=1，有指数$b^i c^j d^k e^l \cdots$的项对应于对i个1，j个2，k个3，l个4等的分划。因此，例如，当n=6时，他实际上以下列顺序给出分划

$$111111, 11112, 1113, 1122, 114, 123, 15, 222, 24, 33, 6 \quad (30)$$

他说明如何递归地列出分划如下(但以与自己的记号相关联的不同语言)：对于k=1, 2, ⋯, n，以k开始，附加其最小部分≥k的$n-k$个(以前列出的)分划。

[我的解]指定发表在会刊上，
同游戏的关系不太大，
但确实包含真理的爱好者所认为的
并非不值得的某些一般的思索。
——阿布拉汉姆·德·莫伊弗勒(1717)

皮·勒·德·蒙特莫特在*Essay d'Analyse sur les Jeux de Hazard* (1708)中，同骰子问题相关联，他制成≤9的数分划成≤6个部分的所有分划表。他的分划以不同于(30)的顺序列出。例如

$$111111, 21111, 2211, 222, 3111, 321, 33, 411, 42, 51, 6 \quad (31)$$

他大概不知道德·莫伊弗勒以前的工作。

迄今为止，几乎还没有任何一个我们已经讨论过的作者真正描述了生成组合模式的步骤。我们只能通过研究他们实际发表的表推导他们的方法，或者说，缺乏他们的方法。而且，在很少的情况下，例如在德·莫伊弗勒明确地描述一个制表方法的论文中，作者假定处理阶为n的情况之前，已经列出了前1, 2, ⋯, $n-1$种情况的所有模式。没有“飞越式”生成模式的方法，如果不考察辅助的表，就不能从一个模式直接移动到它的后继。这实际上已被我们所遇到过的作者所说明，只有克达拉和纳拉燕纳除外。今天的计算机程序员们自然偏爱那些更直接的和只需要少量内存的方法。 68

罗杰尔·约瑟夫·博斯科维奇在*Giornale de' Letterati* (罗马，1747)的393-404页发表了对于分划生成的第一个直接算法，连同面对404页的两个折迭表方法。他的方法产生n=6的分划，输出分别为

$$111111, 11112, 1122, 222, 1113, 123, 33, 114, 24, 15, 6 \quad (32)$$

以精确地同由算法7.2.1.4P所访问分划相反的顺序生成这些分划。他的方法在7.2.1.4节中加以表征。但未指出这样一个事实，即颠倒的顺序比已经选定的顺序稍微容易和更快些。

博斯科维奇在*Giornale de' Letterati*(罗马，1748)，12-27和84-99上发表了连续的两篇论文，以两种方法来推广他的算法。首先，他考虑只生成其部分属于一个给定集合S的分划，使得可以对带有稀疏系数的符号多项式求m次乘幂。(他说，S的所有元素的最大公因式应该是1；而事实上，如果$1 \notin S$，他的方法可能失效)。其次，给定m和n，他介绍了把n分划成m部分的算法。他再次不幸运，完成这个任务的稍微更好的算法7.2.1.4H，随后被发现了，使他失去了成名的机会。

辛登伯格的骗局。算法7.2.1.4H的发明者是卡尔·弗雷德里兹·辛登伯格，他

也重新发现了纳拉燕纳的算法7. 2. 1. 2L，即生成一个多重集合排列的取胜技术。不幸的是，这些小的成功使他认为在数学上作出了革命性的推进——尽管他确实屈尊评论说，像德·莫伊弗勒，欧拉，兰伯特已接近做出类似的发现。

辛登伯格是一个典范的有卓越成就者，他精力极为充沛，如果不是说很有灵感的话。他创立了或协同创立了德国的第一个数学专业杂志，(出版于1786-1789和1794-1800)，而且对于每一个他都贡献了长的论文。他在莱比锡大学好多次担任院长，1792年他还成为那里的校长。如果他是更好的数学家，则德国的数学在莱比锡就会比在柏林或格廷根更为繁荣。

但是他的第一部数学著作*Beschreibung einer ganz neuen Art, nach einem bekannten Gesetze fortgehende Zahlen durch Abzählen oder Abmessen bequem und sicher zu finden* (莱比锡：1776)，充分地预言什么将会出现。他在该小册子中的
69 “全新”思想只对于十进制写出的数的数字组合的重要性。他个人难以相信地通过包含数0000～9999的表的大量折迭页，后边还有分开列出偶数和奇数的两个表(!)来结束他的专著。

辛登伯格发表了表扬他的工作的信件，并邀请他们为他的刊物投稿。1796年，他编辑了*Sammlung combinatorisch-analytischer Abhandlungen*，它的副标题(用德文)指出德·莫弗勒的多项式定理“在所有数学分析中是最重要的命题”。大约有十来人加盟后来成为辛登伯格组合学派，他们发表了充满神秘符号的数以千页的文章，这些必然给许多非数学家以很深的印象。

从计算机科学的观点来看，这个学派的工作不完全是平凡的。例如，亨·奥·罗瑟，(他是辛登伯格最好的学生)注意到，从一个莫斯码序列到词典顺序下的后继或前驱有一个简单的方法。另一个学生约·卡·巴克哈尔特发现，长度为n的莫斯码序列也可通过首先考虑不带折线的，然后带一个折线的，然后带两个折线的等等，容易地生成。他们的动机不像在印度那样，对n个拍子的诗节奏制表，而是列出连续的多项式$K(x_1, x_2, \cdots, x_n)$即等式4.5.3-(4)的项[参见*Archiv für reine und angewandte Mathematik* **1**(1974)，154-194]。而且在上面所引用的辛登伯格1796年的论文集中第53页，乔·西·格鲁吉尔介绍了列出所有排列的一种方法，后来变成为有名的奥德-史密斯算法；参见7.2.1.2节的等式(23)～(26)。

辛登伯格相信，他的方法在标准课程表中值得花费和代数、几何以及微积分等课程相同的时间。但是他和他的信徒们是只作组合列表的组合学家。他们埋头于公式和形式化中，却很少发现有实际兴趣的新数学。尤金·奈特托在莫·康托尔的*Geschichte der Mathematik* **4**(1908)，201-219页已经敬仰地总结了他们的工作“有一个阶段他们控制了德国的市场；然而，他们所挖掘的大多数东西很快陷入不完全值得的忘却中”。

可悲的结果在于一般说来，组合的研究得到了一个坏的名声。戈斯塔·米塔-列夫勒在辛登伯格死后大约100年时间，建立了数学文献的重要图书馆，决定把所有这些著作都放在特殊标记为“颓废者”的架子上。今天在瑞典的米塔-列夫勒研究所的图书馆里仍坚持这个分类，正如该研究所吸引的不是别的，而正是其研究方向为颓废的世界级的组合数学家。

看其光辉的一面，我们可以说，从所有这个活动中至少出现了一本好书。安德列斯·冯·爱丁绍森的*Die Combinatorische Analysis* (维也纳：1826)作为以敏锐的方法来讨论组合生成方法的第一本教材是值得一读的。他在§8中讨论了生成词典顺序的一般原理，并且用它们构造出列举所有排列(§11)、组合(§30)以及分划(§41-§44)的一些好方法。

树在哪里？我们现在已经看到，元组、排列、组合以及分划的列表，在人类历史的相当早的阶段已由感兴趣的和有趣的研究者们搜集了。因此，我们已经在7.2.1.1至7.2.1.5节中解释了所研究的课题的进化，如果我们能追溯在7.2.1.6节中树的生成的根源，我们的故事就完成了。 70

但在计算机发明之前，该课题的历史记录还是一片空白，惟一的例外是19世纪由阿瑟·凯莱所写的一些文章。凯莱关于树的主要文章原来发表于1875年并且重印于*Collected Mathematical Papers*第4卷的427-460页，它以揭示所有有9个或更少未标号顶点的自由树的大量折迭的图作为高峰。早于这篇文章，他也图示了有5个顶点的9个有向树。他用来产生那些表的方法十分复杂，完全不同于算法7.2.1.6O和习题7.2.1.6-90。有多达10个顶点的所有自由树在许多年后由弗·哈拉里和吉·普林斯列出[*Acta Math.* **101**(1958)，158-162]。他还对于没有度数2的顶点或没有对称性的自由树的情况，继续研究到n=12的情况。

然而，为计算机科学家所最为喜爱的树——二叉树、或等价的有序森林、或嵌套的括弧——奇怪地未在参考文献中出现。我们在2.3.4.5节中看到，18和19世纪的许多数学家已经知道如何统计二叉树，我们也知道卡塔兰数C_n枚举几十个不同类型的组合对象。然而，在1950年以前似乎没有任何人以任何这些样子发表阶为4的C_4=14个对象的实际表。这比阶为5的C_5=42个对象少得多。(除非间接地：没有交叉线的(25)中的42个源氏-香框图证实是等价于5节点二叉树和森林。但这个事实在20世纪之前却还未知)。

有一些孤立的情况，往昔的作者们编制了C_3=5个与卡塔兰有关的对象的表。凯莱又是第一个人，在*Philosophical Magazine* **18**(1859)，374-378中示出有3个内节点和4个叶的二叉树如下：

 (33)

(同一文章还图示另外种类的树，等价于所谓的弱序)。于1901年，尤金·奈特托列出在表达式“$a+b+c+d$”中插入括弧的5种方法：

$$(a+b)+(c+d),\ [(a+b)+c]+d,\ [a+(b+c)]+d,\ a+[(b+c)+d],\ a+[b+(c+d)] \tag{34}$$

[*Lehrbuch der Combinatorik* §122。]而鲍罗·厄尔多斯和伊尔文·卡普兰斯基以下列方式列出其部分和为非负的{+1,+1,+1,−1,−1,−1}的5个排列[*Scripta Math.* **12**(1946)，73-75]：

$$1+1+1-1-1-1,\ 1+1-1+1-1-1,\ 1+1-1-1+1-1,\ 1-1+1+1-1-1,\ 1-1+1-1+1-1 \tag{35}$$

尽管只涉及5个对象，我们可以看到，(33)和(34)中的顺序基本上是用尽一切方法
71 的；但只有(35)，它同算法7. 2. 1. 6P匹配，是系统和词典顺序的。

我们还应该简单地说明沃尔瑟·冯·迪克的工作，因为许多近来的工作都以“迪克词”指嵌套括弧的串。除了其他事情之外，迪克是以慕尼黑的德国博物馆的创建者之一而著称的教育家。他关于自由群理论写了两篇先驱性的论文[*Math. Annalen* **20**(1882)，1-44；**22**(1883)，70-108]。但是所谓的迪克词充其量只是同他实际的研究有一个微妙的关系；他研究$\{x_1, x_1^{-1}, \cdots, x_k, x_k^{-1}\}$的词，在重复地删除形如$x_i x_i^{-1}$或$x_i^{-1} x_i$的相邻字母对之后，归结为空的串。仅当我们限制头一种情况$x_i x_i^{-1}$的删除时，才出现了同括弧和树的联系。

因此，我们可以得出结论说，尽管在1950年后才出现了对二叉树和它们的堂兄弟的兴趣的探索，但这样的树代表了其历史根源是相对肤浅的我们的故事的仅有方面。

1950年之后，当然电子计算机的出现改变一切。关于组合生成方法的第一个面向计算机的出版物是由查·布·汤普金斯的文章“变量为排列的问题的机器求解”[*Proc. Symp. Applicd Math.* **6**(1956)，202-205]。随后注定地有千百篇论文出现。

德·亨·莱默所写的多篇论文，特别是他发表在*Proc. Symp. Applied Math.* **10**(1960)的179-193页的“把组合技巧教给计算机”，证明是早期极有影响的论文。[也参见*Proc. 1957 Canadian Math. Congress* (1959)，160-173；*Proc. IBM Scientific Computing Symposium on Combinatorial Problems* (1964)，23-30；以及由埃·福·贝肯巴赫所编的*Applied Combinatorial Mathematics* (Wiley：1964)第1章5-31。]莱默代表了同以前几代人的重要的联系。例如，斯坦福大学的图书馆记录表明，他于1932年1月份曾检查奈特托的*Lehrbuch der Combinatorik*。

我们已经在以前几节中引述了同我们已经研究的特定算法有关的主要著作，因此在这里没有必要重复。但是，首先把关于这个课题的部分以一致的框架收集在一起的课本和专著也很重要。特别是，有三本书，就它们建立的一般原理来说，值得一提：

- 马克·布·威尔斯的*Elements of Combinatorial Computing* (Pergamon Press，1971)，特别是第5章。
- 阿尔伯特·尼珍豪斯和希尔伯特·威尔弗的*Combinatorial Algorithms* (Academic Press. 1975)。第二版发表于1978年，它包含了补充内容，威尔弗随后写了*Combinatorial Algorithms An Update* (费城：SIAM，1989)。
- 由爱德华·马·莱因戈尔德，尤尔格·尼耶弗尔吉特以及纳尔辛格迪奥所写的*Combinatorial Algorithms：Theory and Practice* (Prentice-Hall，1977)，特别是第5 章的内容。

罗伯特·塞德杰维克在*Computing Surveys* **9**(1977)，137-164，314编辑了关于排列生成方法的第1本广泛的评述。卡尔拉·沙瓦吉在*SIAM Review* **39**(1997)，605-
72 629上关于格雷码的评述文章是另一个里程碑。

我们上边指出过，在计算机程序员产生对于生成由卡塔兰计数对象的算法的兴趣之前，这些算法还未发现。最初的一些要发表的这些算法在7.2.1.6节中未曾提及，因为它们已被更好的技术所超越。但是，在这里列出它们是适当的。首先，我们引述了关于生成有向树的文章*Machine Intellgence* **3**(1968)，43-60。胡·伊·斯戈因斯在同一篇文章中给出了生成有序树的两个递归算法。他的算法讨论了表示为二进位串的二叉树，它们实际上等价于波兰前缀表示或嵌套的括弧。然后，在上面所引用的他的书的5.5.4节中，马克·威尔斯通过把它们表示为不交叉的集合分划来生成二叉树。而且，加里·克诺特[*CACM* **20**(1977)，113-115]给出了二叉树的递归排阶和去阶算法，并且通过表7.2.1.6-3的中根顺序到前根顺序的排列 $q_1 \cdots\ q_n$ 表示它们。

自20世纪50年代以来许多作者发表了为生成一个给定图的所有生成树的算法，它们起源于对电子网络研究的启示。在最初的这些文章当中，有中川敬之，*IRE Trans*. CT-5 (1958)，122-127；前田渡，*IRE Trans*. CT-6(1959)。136-137，394；渡部和，*IRE Trans*. CT-7(1960)，296-302；塞·哈基米，*J. Franklin Institute* **272**(1961)，347-359。

近来关于整个课题的导论可以在唐纳德·劳·克里赫和道格拉斯·罗·斯廷森的*Combinatorial Algorithms：Generation，Enumeration，and Search* (CRC Rress，1999)的第2章和第3章中找到。

弗朗克·拉斯利亚在准备写一本标题为*Combinatorial Generation*的书，将包含详尽的讨论以及广泛的参考文献。他已经写成了好几章草稿，可在互联网上找到。

习　题

下列习题中，有许多题请求现代读者找出和/或改正以往文献中的错误。关键点不在于为21世纪的我们如何聪明而洋洋得意；而在于理解，即使是这一学科的先驱者也可能出错。学习一套思想的一个好方法实际上不像今天的计算机科学家和数学家看起来那么简单，而是去发现，当这些思想还是新的时，某些世界领先的思想家不得不为这些概念进行奋斗的地方。

1. *[15]* 在易经中出现“计算”的思想吗？

▶2. *[M30]* (基因码。)DNA分子是4个字母{T，C，A，G}表的“核”串，而且大多数的蛋白质分子是20个字母的字母表{A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y}的“氨基酸”串。三个连续的核xyz形成一个“密码子”，DNA的一个串线$x_1\,y_1\,z_1\,x_2\,y_2\,z_2\cdots$确定蛋白质$f\ (x_1 y_1 z_1) f\,(x_2 y_2 z_2)\cdots$，其中$f\,(x, y, z)$是在数组

$$\begin{pmatrix} F & S & Y & C \\ F & S & Y & C \\ L & S & - & - \\ L & S & - & W \end{pmatrix} \begin{pmatrix} L & P & H & R \\ L & P & H & R \\ L & P & Q & R \\ L & P & Q & R \end{pmatrix} \begin{pmatrix} I & T & N & S \\ I & T & N & S \\ I & T & K & R \\ M & T & K & R \end{pmatrix} \begin{pmatrix} V & A & D & G \\ V & A & D & G \\ V & A & E & G \\ V & A & E & G \end{pmatrix}$$

中矩阵x的z行y列元素。这里(T，C，A，G)=(1, 2, 3, 4)。例如，f (CAT)是矩阵2的1行3列的元素，即H。编码继续进行直到一个密码子导致一个停止符“—”为止。 73

a) 证明有一个把每个密码子映射到易经的6线形的简单方法，有性质：21个可能的结果{A, C, D, ⋯, W, Y, —}对应于周文王顺序(1)的21个连续的6线形。

b) 这是一个有意义的发现吗？

3. [*20*] 在类似于(2)的协词典顺序下，什么是有30拍的第一百万诗步？⌣⌣⌣—⌣——⌣⌣⌣⌣——⌣⌣⌣⌣⌣⌣⌣—⌣—的阶是什么？

4. [*19*] 试分析表1中的多诺罗排列表的缺点。

5. [*16*] 在(7)中的5个音符排列的克彻尔表错在哪儿？

6. [*25*] 梅森在*Traitez de la Voix et des Chants* (1636)一书的108-110页发表了前64个阶乘的表。对于64!的值他认为$\approx 2.2\times 10^{89}$；但它应该是$\approx 1.3\times 10^{89}$。试找到他的书并尝试看看他错在何处？

7. [*20*] 按照关孝和的规则(8)和(9)，{1, 2, 3, 4, 5}的排列中哪些是“活的”，哪些是“死的”？

▶8. [*M27*] 对(9)做一个修正，使关孝和步骤成为正确的。

9. [*15*] 从(11)，推导出写阿拉伯数字(0, 1, ⋯, 9)的阿拉伯方式。

▶10. [*HM27*] 在鲁杜斯 · 克勒里卡利斯的思想下，预期在获得所有可能的美德之前，投掷3个骰子的期望次数是多少？

11. [*21*] 对图45右边的勒鲁尔垂直表进行解密。它代表哪20个组合的对象？提示：不要为打字错误所误导。

12. [*M20*] 把斯希林格尔的万有循环(13)同习题7. 2. 1. 3-106中的泊因索特万有循环相关联。

13. [*21*] 施库廷应该写什么以代替(17)？再给出对于多重集合{a, a, a, b, b, c}的组合的对应表。

▶14. [*20*] 从伊泽奎尔多的*De Combinatoine* 的§95，完成下列序列

ABC ABD ABE ACD ACE ACB ADE ADB ADC AEB ⋯

15. [*15*] 如果以词典顺序列出允许重复的{1, ⋯, m}的所有n个组合，以数j开始的有多少？

16. [*20*] (纳拉燕纳 · 潘迪塔，1356。)试设计一个算法来生成n分成为$\leqslant q$部分的所有组成，即所有有序的分划$n=a_1+\cdots+a_t$。其中，对于$1\leqslant j\leqslant t$，$1\leqslant a_j\leqslant q$，而且$t$是任意的。当$n$=7和$q$=3时，展示你的方法。

17. [*HM27*] 试分析习题15的算法。

18. [*10*] 巧妙问题：莱布尼茨于1666年发表*Dissertatio de Arte Combinatoria*。为什么这是一个特别吉祥的年份？是排列式的年吗？

19. [*17*] 在普提亚尼尤斯的诗(20)中的哪一个“tibi”被处理为⌣— 而不是⌣⌣？

20. [*M25*] 为纪念3名卓越的贵族于1617年对德里斯顿的访问，一位诗人发表了六步诗的1617个排列

Dant tria jam Dresdae, ceu sol dat, lumena lucem。

“三位大人现在来到德罗斯顿，犹如太阳把光亮送给光芒地方”。[格里高尔 · 克里皮斯，*Proteus Poeticus* (Leipzig：1617)。] 有多少这些词的排列实际上恰当地符合格律？提示：这首诗在第1个和第5个诗步中有扬抑抑格，在其他处有扬扬格。

74

21. [*HM30*] 当$p+q+r=s+2t$时，令$f(p, q, r; s, t)$是通过连接串$\{s\cdot o,\ t\cdot oo\}$形成(o^p, o^q, o^r)的方式数。例如，$f(2, 3, 2; 3, 2)=5$，因为5种方式是

$$(o|o, o|oo, oo),\quad (o|o, oo|o, oo),\quad (oo, o|o|o, oo),\quad (oo, o|oo, o|o),\quad (oo, oo|o, o|o).$$

a) 试证明$f(p, q, r; s, t)=[u^p v^q w^r z^s]1/((1-zu-u^2)(1-zv-v^2)(1-zw-w^2))$。

b) 在第5个诗步不从词的中间开始这一附加条件下，使用函数f来枚举(19)的可符合格律的排列。

c) 现在枚举剩下的情况。

▶22. [*M40*] 查找由普列斯特，沃利斯，惠特沃思和哈特里发表的关于tot-tibi问题原来的讨论。他们犯了什么错误？

23. [*20*] 52个源氏－香框图中的什么顺序对应于算法7. 2. 1. 5H？

▶24. [*23*] 19世纪早期，本田利明给出了生成$\{1,\cdots,n\}$的所有分划的一个递归规则。当$n=4$时，他的算法以以下顺序产生它们：



你能猜出当$n=5$时对应的顺序是什么吗？提示：参见(26)。

25. [*15*] 16世纪*The Arte of English Poesie*的作者仅对在习题7. 2. 1. 5-35的意义下“完备”的韵律方案感兴趣。换言之，每行至少应同隔一行押韵。而且，在习题7. 2. 1. 2-100的意义下，这个方案应该是“不可分解的”：像12|345这样的分划可分解成2行诗后边接3行诗。这个方案不应平凡地由全都彼此押韵的行组成。在这些条件下，(28)是5行韵律方案的一个完备的表吗？

▶26. [*HM25*] 有多少n行韵律方案满足习题25的限制条件？

▶27. [*HM31*] 集合分划14|25|36可由一个源氏-香框图表示为，但对于这个分划，每个这样的框图必须至少有三处行交叉，而交叉线有时是非期望的。$\{1,\cdots,n\}$有多少个分划，其源氏-香框图中至多只有一个交叉？

▶28. [*25*] 令a,b和c为素数。约翰 · 沃利斯列出了a^3b^2c的所有可能因式分解如下：

$cbbaaa$, $cbbaa\cdot a$, $bcaaa\cdot b$, $bbaaa\cdot c$, $cbba\cdot aa$, $cbba\cdot a\cdot a$, $cbaa\cdot ba$,
$cbaa\cdot b\cdot a$, $bbaa\cdot ca$, $bbaa\cdot c\cdot a$, $caaa\cdot bb$, $caaa\cdot b\cdot b$, $baaa\cdot cb$, $baaa\cdot c\cdot b$, $cbb\cdot aaa$,
$cbb\cdot aa\cdot a$, $cbb\cdot a\cdot a\cdot a$, $cba\cdot baa$, $cba\cdot ba\cdot a$, $cba\cdot aa\cdot b$, $cba\cdot b\cdot a\cdot a$, $bba\cdot caa$, $bba\cdot ca\cdot a$,
$bba\cdot aa\cdot c$, $bba\cdot c\cdot a\cdot a$, $caa\cdot bb\cdot a$, $caa\cdot ba\cdot b$, $caa\cdot b\cdot b\cdot a$, $baa\cdot cb\cdot a$, $baa\cdot ca\cdot b$, $baa\cdot ba\cdot c$,
$baa\cdot c\cdot b\cdot a$, $aaa\cdot cb\cdot b$, $aaa\cdot bb\cdot c$, $aaa\cdot c\cdot b\cdot b$, $cb\cdot ba\cdot aa$, $cb\cdot ba\cdot a\cdot a$, $cb\cdot aa\cdot b\cdot a$,
$cb\cdot b\cdot a\cdot a\cdot a$, $bb\cdot ca\cdot aa$, $bb\cdot ca\cdot a\cdot a$, $bb\cdot aa\cdot c\cdot a$, $bb\cdot c\cdot a\cdot a\cdot a$, $ca\cdot ba\cdot ba$, $ca\cdot ba\cdot b\cdot a$,
$ca\cdot aa\cdot b\cdot b$, $ca\cdot b\cdot b\cdot a\cdot a$, $ba\cdot ba\cdot c\cdot a$, $ba\cdot aa\cdot c\cdot b$, $ba\cdot c\cdot b\cdot a\cdot a$, $aa\cdot c\cdot b\cdot b\cdot a$,
$c\cdot b\cdot b\cdot a\cdot a\cdot a$.

他使用什么算法按这个顺序生成它们？

▶29. [*24*] 沃利斯以什么顺序生成数$abcde=5\cdot7\cdot11\cdot13\cdot17$的所有因子分解？把你的答案作为源氏－香框图的序列给出。

30. [*M20*] 在$(a_0z+a_1z^2+a_2z^3+\cdots)^m$中$a_1^{i_1}a_2^{i_2}\cdots z^{m+n}$的系数是什么？(参见(29)。)

31. [*20*] 试对分划的莫 · 伊弗勒和德 · 蒙特莫特顺序即(30)和(31)同算法7.2.1.4P做比较。

32. [*21*] (普 · 约 · 博斯科维奇，1748。)列出20的所有分划，即其所有部分为1，7或10。并设计列出任何给定整数$n>0$的所有这样分划的表的算法。 75

习题答案

7.2.1.6 节

1. 它能“看见”每个内部节点左边的一个左括弧和每个内部节点底部的右括弧。或者，它能结合右括弧和它所遇到的外部节点——非常后面的□例外。参见习题20。

2. Z1. [初始化。] 对于$0\leqslant k\leqslant n$，置$z_k\leftarrow 2k-1$(假设$n\geqslant 2$)。

Z2. [访问。] 访问树组合$z_1 z_2 \cdots z_n$。

Z3. [是容易情况吗？] 如果$z_{n-1}<z_{n-1}$，则置$z_n\leftarrow z_n-1$，并返回Z2。

Z4. [求j。] 置$j\leftarrow n-1$和$z_n\leftarrow 2n-1$。当$z_{j-1}=z_j-1$时，置$z_j\leftarrow 2j-1$和$j\leftarrow j-1$。

Z5. [减小z_j。] 如果$j=1$，则算法终止。否则置$z_j\leftarrow z_j-1$，并返回到Z2。▮

3. 以前根顺序对森林的节点标号。$a_1\cdots a_{2n}$的前z_k-1个元素包含$k-1$个左括弧和z_k-k个右括弧。所以当“蚯蚓”先抵达节点k时，左括弧比右括弧超出$2k-1-z_k$个，而$2k-1-z_k$是该节点的层(或深度)。

令$q_1\cdots q_n$是$p_1\cdots p_n$的逆，使节点k是在后根顺序的第q_k个节点。由于k出现在$p_1\cdots p_n$中j的左边，当且仅当$q_k<q_j$时，我们看到c_k是在前根顺序k前边但在后根顺序在它之后的节点j的个数，即k的祖先的个数；又是k的层。

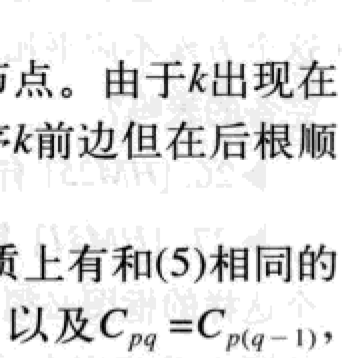
另一个证明：我们可以证明，两个序列$z_1\cdots z_n$和$c_1\cdots c_n$实质上有和(5)相同的递归结构：当$0\leqslant p\leqslant q$时，$Z_{pq}=(Z_{p\,(q-1)}+1^p)$，$1(Z_{(p-1)q}+1^{p-1})$；以及$C_{pq}=C_{p(q-1)}$，$(q-p)C_{(p-1)q}$。(考虑最后、次最后等左括弧的配偶。)

顺便提及，公式‘$c_{k+1}+d_k=c_k+1$’等价于(11)。

4. 几乎是真的。但$d_1\cdots d_n$和$z_1\cdots z_n$以递减顺序出现，而$p_1\cdots p_n$和$c_1\cdots c_n$以递增顺序出现。(排列的序列$p_1\cdots p_n$的词典顺序性质并不自动由继承对应的反演表$c_1\cdots c_n$的词典顺序得到，但对于$p_1\cdots p_n$的特殊类此结果确实成立。)

5. $d_1\cdots d_{15}$=020020010320104；$z_1\cdots z_{15}$=1 2 5 6 7 10 11 12 14 15 19 22 23 25 26；$p_1\cdots p_{15}$=2 1 5 4 8 10 9 7 11 6 13 15 14 12 3；$c_1\cdots c_{15}$=0 1 0 1 2 1 2 3 3 4 2 1 2 2 3。

6. 和通常一样匹配括弧，然后简单地向上和左右弯曲串直到a_{2n}同a_1相邻为止。注意左括弧和右括弧的区别可以从上下文中重新构造出来。令a_1对应于圆的底部，如表1所示，产生所示的框图。[阿 · 厄里拉(A. Errera)，*Mémoires de la Classe Sci 8°Acad. Royale de Belgique* (2)，11，6(1931)，26页。]



7. (a)它等于))( )⋯( )；置$a_1\leftarrow$ ‘(’ 将恢复初始的串。(b)初始的二叉树(从步骤B1)将已被恢复，除非$l_n=n+1$。

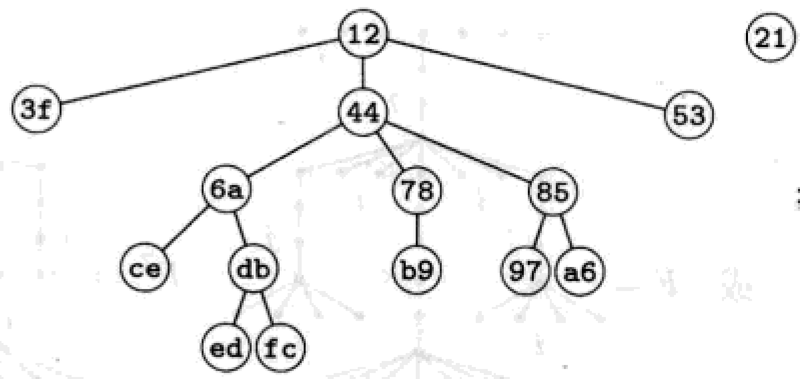
8. $l_1\cdots l_{15}$=2 0 4 5 0 7 8 0 10 00 13 0 15 0；$r_1\cdots r_{15}$=3 0 0 6 0 12 11 9 0 0 0 0 14 0 0；$e_1\cdots e_{15}$=1 0 3 1 0 2 2 0 1 0 0 2 0 1 0；$s_1\cdots s_{15}$=1 0 12 1 0 5 3 0 1 0 0 3 0 1 0。

9. 节点j是节点k的祖先，当且仅当$s_j+j\geqslant k$。(结果，有$c_1+\cdots+c_n=s_1+\cdots+s_n$)。

10. 如果j是第k个左括弧的下标z_k，我们有$w_j=c_k+1$和$w_{j'}=c_k$，其中j'是匹配右括弧的下标。 76

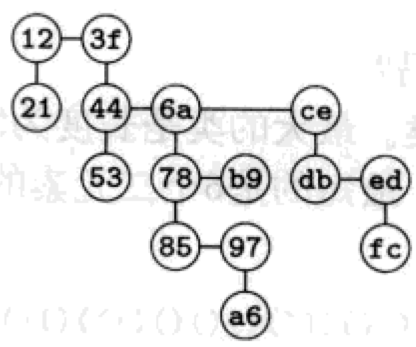
11. 在$a_{2n}\cdots a_1$中交换左和右括弧以得到$a_1\cdots a_{2n}$的镜面映象。

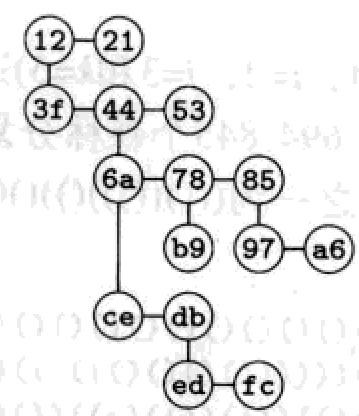
12. (4)的镜面反射对应于森林



;

如果水平地和垂直地画右兄弟和左儿子，则转置的重要性是更清晰的森林式的，然后作一个矩阵式的转置。





13. (a) 通过对节点个数运用归纳法，我们有前根顺序(F^R)=后根顺序$(F)^R$，和后根顺序(F^R)=前根顺序$(F)^R$。

(b) 令F对应于二叉树B，于是前根顺序(F)=前根顺序(B)，且后根顺序(F)=中根顺序(B)。如2.3.2-(6)之后所说明的那样。因此，前根顺序(F^T)=前根顺序(B^R)=后根顺序$(B)^R$同前根顺序(F)或后根顺序(F)都没有简单关系但后根顺序(F^T)=中根顺序(B^R)=中根顺序$(B)^R$=后根顺序$(F)^R$。

14. 按照答案13，当F自然地对应于B时，后根顺序(F^{RT})=前根顺序(F)=前根顺序(B)；而且后根顺序(F^{TR})=前根顺序$(F^T)^R$=后根顺序(B)。因此，等式$F^{RT}=F^{TR}$成立当且仅当F至多有一个节点时。

15. 如果F^R自然地对应于二叉树B'，则B'的根是F的最右树的根。在B'中节点x的左链接是F^R中x最左的儿子，它是F中x的最右儿子。类似地，右链接是F中x的左兄弟的右链接。

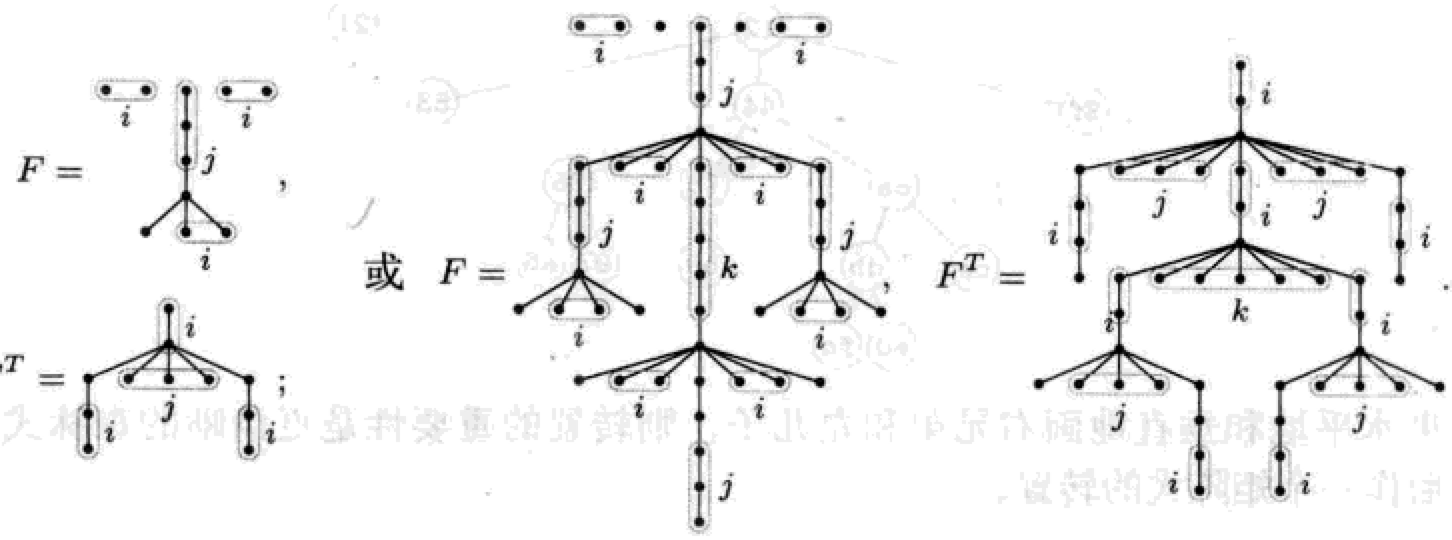
注意：由于B自然地对应于F^{RT}，答案13告诉我们，中根顺序(B)=后根顺序(F^{RT})=后根顺序$(F^R)^R$=前根顺序(F)。

16. 通过把F的树放置在后根顺序中G的第一个节点之下得到森林$F\mid G$。由于$F\mid(G\mid H)=(H^T G^T F^T)^T=(F\mid G)\mid H$，因此结合律成立。顺便指出，注意后根顺序$(F\mid G)$=后根顺序$(F)$后根顺序$(G)$，而且当$G$非空时，$F\mid(GH)=(F\mid G)H$。

17. 任何非空的森林都可写成$F=(G\mid\cdot)H$，其中·表示1节点森林。于是，$F^R=H^R(G^R\mid\cdot)$，且$F^T=(H^T\mid\cdot)G^T$。特别是，除非H是空森林Λ，否则我们不能有$F^R=F^T$，

因为H^R的第一棵树不能是$H^T|\cdot$；而G也必是Λ。其次，$F=F^T$当且仅当$G=H^T$。在此情
77 况下，除非$G=\Lambda$，否则我们也不能有$F^R=F^{RT}$。G^{TR}的第一棵树有比G本身更多的节点。

除非$F=F^R$，否则我们不能有$F^{RT}=F^{TR}$，这显然为真。在这个假设下，$F^{RT}=F^{TR}$当且仅当F和F^T都是自共轭的。戴维·卡兰发现这样的森林的两个无穷族，有参数$i, j, k\geqslant 0$：



(在这些例子中，$i=2$，$j=3$和$k=5$)还有其他情况吗？

18. C_{15}=9 694 845个森林分划成20 982个类。最大的类是长度为58 968的一个循环，其元素之一为((()(()))())()((()(())())())()。最短的是6个二元素的类(对应于习题17)。由

()()()()()()()()()()()()()()(),　　()()()()((()()()()()))()()()(),
()()()((((()()()()))))()()(),　　()()((((((((()()())))))))))()(),
()((()())((()(())()))(()()))(),　　()(((((((((((()())))))))))))(),

以及它们的转置组成。稍微奇怪的串(((((((())))))))()()()()()()()、()()()()()()()(((((((())))))))以及((((((((()()()()()()()))))))))，每个都有楔形的二叉树，并且形成大小为3的惟一类。从()((()(()()))(())((()())()))()到((()())(()())(())(()())(()()))的通路有3120个元素，其中之一为(2)。按照答案19中的猜测，最短可能的循环长度为6；当$n=15$时，有66个这样的环(次最短循环是惟一的，有长度10，包括()(()()())()((((())()))(()))。

19. 由算法P从F_j到F_{j+1}的转换可释意如下：“求前根顺序最后的节点，比如说x，它有一个左兄弟y。从其家族中删除x，并使它成为y的新的最右儿子。如果$x<n$，就把x的所有后裔$x+1,\cdots,n$变成为平凡的1节点树”。

把F_j^R变成为F_{j+1}^R的转换因此可表述如下：如果我们回想起，在前根顺序下F_j的第k个节点是后根顺序下F_j^R的从后数第k个节点：“求前根顺序的第一个节点x，它有一个右兄弟y，从其家族中删去x并使它成为y的新的最左儿子。如果$x<n$，则把x的所有后裔$x+1,\cdots,n$变成平凡的1节点树。”

类似地，我们可以释意由算法B所描述的从G_j到G_{j+1}的变换：“求j，即最左的非平凡树的根。然后求k，即它最右的儿子。从j的家族中删除k及它的后裔，把它们插入j和j的右兄弟之间。最后，如果$j>1$，使j和它的兄弟全都成为$j-1$的儿子，而且$j-1$成为$j-2$的一个儿子，等等。”

当这个变换把左兄弟/右儿子表示从 G_j^{RT} 改变成为 G_{j+1}^{RT} 时(参见习题15)，结果是它等于在左儿子/右兄弟表示中把 F_j^R 变成为 F_{j+1}^R的变换。因此，$G_j^{RT}=F_j^R$，因为当j=1时这个等式显然成立。 78

（由此得出，对于由算法B生成的二叉树表的序列$e_1\cdots e_{n-1}$，恰巧是由算法P生成的括弧串的表序列$d_{n-1}\cdots d_1$。这个现象在表1和表2中示出。）

森林F^{RTR}称为F的对偶，参见习题26(f)。在森林的表之间的若干对称性已由余明照所剖析。*Comp. J.* **32**(1989)，76-85。

20. (a)这一断言推广了引理2.3.1P，可容易地通过归纳法证明。

(b)下列过程事实上几乎和算法P相同：

T1. [初始化。] 对于$1\leqslant k\leqslant n$，置$b_{3k-2}\leftarrow 3$和$b_{3k-1}\leftarrow b_{3k}\leftarrow 0$；并置$b_0\leftarrow b_N\leftarrow 0$和$m\leftarrow N-3$，其中$N=3n+1$。

T2. [访问。] 访问$b_1\cdots b_N$。(现在$b_m=3$，$b_{m+1}\cdots b_N=0\cdots 0$。)

T3. [是容易情况吗？] 置$b_m\leftarrow 0$。如果$b_{m-1}=0$，则置$b_{m-1}\leftarrow 3$，$m\leftarrow m-1$，转到T2。

T4. [求j。] 置$j\leftarrow m-1$，和$k\leftarrow N-3$。当$b_j=3$时，置$b_j\leftarrow 0$，$b_k\leftarrow 3$，$j\leftarrow j-1$和$k\leftarrow k-3$。

T5. [增加b_j。] 如果j=0则算法终止。否则，置$b_j\leftarrow 3$，$m\leftarrow N-3$，并返回T2。 ▌

[参见施·扎克斯，*Theoretical Comp. Sci.* **10**(1980)，63-82。在该论文中，扎克斯指出，利用实际上等于习题2的答案的算法，生成使得$b_j=3$的下标j的序列$z_1\cdots z_n$甚至更容易，因为一个正确的三叉树组合$z_1\cdots z_n$是以不等式$z_{k-1}<z_k\leqslant 3k-2$表征的。]

21. 对于这个问题，我们实际上能够组合算法P和算法7.2.1.2L。为方便起见，我们将假定$n_t>0$和$n_1+\cdots+n_t>1$。

G1. [初始化。] 置$l\leftarrow N$。对于$j=t,\cdots,2,1$(以此顺序)，做下列操作n_j次：置$b_{l-j}\leftarrow j$，$b_{l-j+1}\leftarrow\cdots\leftarrow b_{l-1}\leftarrow 0$，以及$l\leftarrow l-j$。最后置$b_0\leftarrow b_N\leftarrow c_0\leftarrow 0$，以及$m\leftarrow N-t$。

G2. [访问。] 访问$b_1\cdots b_N$。(这时$b_m>0$且$b_{m+1}=\cdots=b_N=0$。)

G3. [是容易的情况吗？] 如果$b_{m-1}=0$，则置$b_{m-1}\leftarrow b_m$，$b_m\leftarrow 0$，$m\leftarrow m-1$并返回G2。

G4. [求j。] 置$c_1\leftarrow b_m$，$b_m\leftarrow 0$，$j\leftarrow m-1$，以及$k\leftarrow 1$。当$b_j\geqslant c_k$时，置$k\leftarrow k+1$，$c_k\leftarrow b_j$，$b_j\leftarrow 0$，并且$j\leftarrow j-1$。

G5. [增加b_j。] 如果$b_j>0$，则求使得$b_j<c_l$的最小的$l\geqslant 1$，并且交换$b_j\leftrightarrow c_l$。否则，如果$j>0$，则置$b_j\leftarrow c_1$，并且$c_1\leftarrow 0$。否则终止。

G6. [颠倒和延展。] 置$j\leftarrow k$和$l\leftarrow N$。当$c_j>0$时，置$b_{l-c_j}\leftarrow c_j$，$l\leftarrow l-c_j$，以及$j\leftarrow j-1$。然后置$m\leftarrow N-c_k$，并返回G2。 ▌

这个算法假定$N>n_1+2n_2+\cdots+tn_t$。[参见*SICOMP* **8**(1979)，73-81。]

22. 首先注意d_1可能增加当且仅当链接表示中$r_1=0$。否则，$d_1\cdots d_{n-1}$的后继通过寻找有$d_j>0$的最小的j和置$d_j\leftarrow 0$，$d_{j+1}\leftarrow d_{j+1}+1$来得到。我们可以假定$n>2$。

K1. [初始化。] 对于$1\leqslant k<n$，置$l_k\leftarrow k+1$，以及$r_k\leftarrow 0$，并置$l_n\leftarrow r_n\leftarrow 0$。

K2. [访问。] 访问由$l_1 l_2 \cdots l_n$和$r_1 r_2 \cdots r_n$表示的二叉树。

K3. [是容易情况吗？] 置$y \leftarrow r_1$。如果$y=0$，则置$r_1 \leftarrow 2$，$l_1 \leftarrow 0$，并返回到K2。否则，如果$l_1=0$，则置$l_1 \leftarrow 2$，$r_1 \leftarrow r_2$，$r_2 \leftarrow l_2$，$l_2 \leftarrow 0$，并返回到K2。否则，置$j \leftarrow 2$和$k \leftarrow 1$。

79 K4. [求j和k。] 如果$r_j>0$，则置$k \leftarrow j$和$y \leftarrow r_j$。如果$j \neq y-1$，则置$j \leftarrow j+1$并重复这个步骤。

K5. [洗子树。] 置$l_j \leftarrow y$，$r_j \leftarrow r_y$，$r_y \leftarrow l_y$，而且$l_y \leftarrow 0$。如果$j=k$，转到K2。

K6. [移动子树。] 如果$y=n$，则终止。否则，当$k>1$时，置$k \leftarrow k-1$，$j \leftarrow j-1$，和$r_j \leftarrow r_k$。然后，当$j>1$时，置$j \leftarrow j-1$和$r_j \leftarrow 0$，返回到K2。 ▮

[参见习题45中的分析。柯尔施(*Comp*. J. **48**(2005))，488-497]已经证明这个算法可以以一个有趣的方式推广到t叉树；而且他还找到算法B的一个有效的t叉推广。

23. (a) 由于z_n在$2n-1$开始向后和向前C_{n-1}次，当$n>1$时，它结束于$2n-1-(C_{n-1} \bmod 2)$。而且，对于所有$n \geqslant j$，z_j最后的值是常数。因此，最后的串$z_1 z_2 \cdots$是1 2 5 6 9 11 13 14 17 19⋯，包含除3, 7, 15, 31，⋯之外所有$<2n$的奇数。

(b) 类似地，表征最后树的前根顺序排列为$2^k\ 2^{k-1}\ \cdots\ 1\ 3\ 5\ 6\ 7\ 9\ 10 \cdots$，其中$k=\lfloor \lg n \rfloor$。森林式的对于$1<j \leqslant k$，节点$2^j$是$2^{j-1}$个节点$\{2^{j-1}, 2^{j-1}+1, \cdots, 2^j-1\}$的父亲，而且树$\{2^k+1, \cdots, n\}$是平凡的。

注意：如果在算法N已经结束之后，在步骤N2再启动，它将生成相同的序列，但是向后的。算法L有相同的性质。

24. $l_0 l_1 \cdots l_{15}$=2 0 1 0 3 0 0 6 5 0 8 0 0 12 11 4；$r_1 \cdots r_{15}$=0 15 0 10 7 0 0 9 0 14 13 0 0 0 0；$k_1 \cdots k_{15}$=0 0 2 2 4 5 5 4 8 4 10 11 11 10 2；$q_1 \cdots q_{15}$=2 1 15 4 3 10 8 5 7 6 9 14 11 13 12；$u_1 \cdots u_{15}$=12 3 1 0 0 5 0 3 1 0 0 1 0 1 0。(如果对森林F的节点以后根顺序编号，k_j是j的左兄弟；或者，如果j是p的最左儿子，$k_j=k_p$；换个方法来说，k_j是在森林F^{TR}中j的父亲。而且k_j也是$j-1-u_{n+1-j}$，即小于j的$q_1 \cdots q_n$中j左边元素的个数。)

25. 从算法N和R获得一个提示，要把每棵$n-1$节点的树推广成两棵或多棵n节点树的一个表。此情况的思想是使n成为在每个这样的表的开始和结尾的二叉树中的$n-1$的一个儿子。以下算法使用另外的链接字段p_j和s_j，其中p_j指向在森林中j的父亲，而s_j指向j的左兄弟或指向j的最右兄弟，如果j是在它的家族中最左的话。(当然这些指针p_j和s_j和表1中的排列$p_1 \cdots p_n$或表2中的范围坐标$s_1 \cdots s_n$不相同。事实上，$s_1 \cdots s_n$是以下习题33的排列λ。)

M1. [初始化。] 对于$1 \leqslant j \leqslant n$，置$l_j \leftarrow j+1$，$r_j \leftarrow 0$，$s_j \leftarrow j$，$p_j \leftarrow j-1$和$o_j \leftarrow -1$，除非$l_n \leftarrow 0$。

M2. [访问。] 访问$l_1 \cdots l_n$和$r_1 \cdots r_n$。然后置$j \leftarrow n$。

M3. [求j。] 如果$o_j>0$，则置$k \leftarrow p_j$，而且，如果$k \neq j-1$则转向M5。如果$o_j<0$，则置$k \leftarrow s_j$并且，如果$k \neq j-1$则转向M4。如果在两种情况下都有$k=j-1$，则置$o_j \leftarrow -o_j$，$j \leftarrow j-1$，重复这一步。

M4. [向下转移。] (这时k是j的左兄弟，或者j的家族的最右成员)。如果$k \geqslant j$，再如果$j=1$就终止；否则置$x \leftarrow p_j$，$l_x \leftarrow 0$，$z \leftarrow k$以及$k \leftarrow 0$(由此把节点j从它的父亲和顶层的标题拆开)。如果$k<j$，则置$x \leftarrow p_j+1$，$z \leftarrow s_x$，$r_k \leftarrow 0$和$s_x \leftarrow$

k(由此从k拆开节点j并下降一层)。然后，置$x \leftarrow k+1$，$y \leftarrow s_x$，$s_x \leftarrow z$，$s_j \leftarrow y$，$r_y \leftarrow j$以及$x \leftarrow j$。当$x \neq 0$时，置$p_x \leftarrow k$以及$x \leftarrow r_x$，返回到M2。

M5. [向上转移。](这时k是j的父亲)。置$x \leftarrow k+1$，$y \leftarrow s_j$，$z \leftarrow s_x$，$s_x \leftarrow y$，而且$r_y \leftarrow 0$。如果$k \neq 0$，则置$y \leftarrow p_k$，$r_k \leftarrow j$，$s_j \leftarrow k$，$s_{y+1} \leftarrow z$，以及$x \leftarrow j$；否则，置$y \leftarrow j-1$，$l_y \leftarrow j$，$s_j \leftarrow z$，和$x \leftarrow j$。当$x \neq 0$时，置$p_x \leftarrow y$并且$x \leftarrow r_x$，返回到M2。▌ 80

运行时间注释：在习题44中我们可以论断，步骤M3花费$2C_n+3(C_{n-1}+\cdots+C_1)$次内存访问，而步骤M4和M5一起花费$8C_n-2(C_{n-1}+\cdots+C_1)$加上$x \leftarrow r_x$次数的两倍。这后一个量难以精确地分析。例如，当$n=15$和$j=6$时，算法在分别为(45, 23, 7, 9, 2, 4)的情况下，共精确地置$x \leftarrow r_x$ (1, 2, 3, 4, 5, 6)次。但当j给定时，启发式地探索$x \leftarrow r_x$的平均次数应当近似地为$2-2^{j-n}$次，因此整个大约为$(2C_n-(C_n-C_{n-1})-(C_{n-1}-C_{n-2})/2-(C_{n-2}-C_{n-3})/4-\cdots)/C_n \approx 8/7$。经验检测证实这个预期的行为，说明当$n \to \infty$时每个树的总费用接近于$265/12 \approx 12.6$次内存访问。

26. (a) 这个条件显然是必要的。如果它成立，就能惟一地构造F：节点1和它的兄弟是森林的根，而它们的后裔由非交叉的分划递归地定义。(事实上，我们可以从Π的限制增长序列$a_1 \cdots a_n$直接计算深度坐标$c_1 \cdots c_n$：置$c_1 \leftarrow 0$和$i_0 \leftarrow 0$。对于$2 \leqslant j \leqslant n$，如果$a_j > \max(a_1, \cdots, a_{j-1})$，置$c_j \leftarrow c_{j-1}+1$，以及$i_{a_j} \leftarrow c_j$，否则置$c_j \leftarrow i_{a_j}$。)

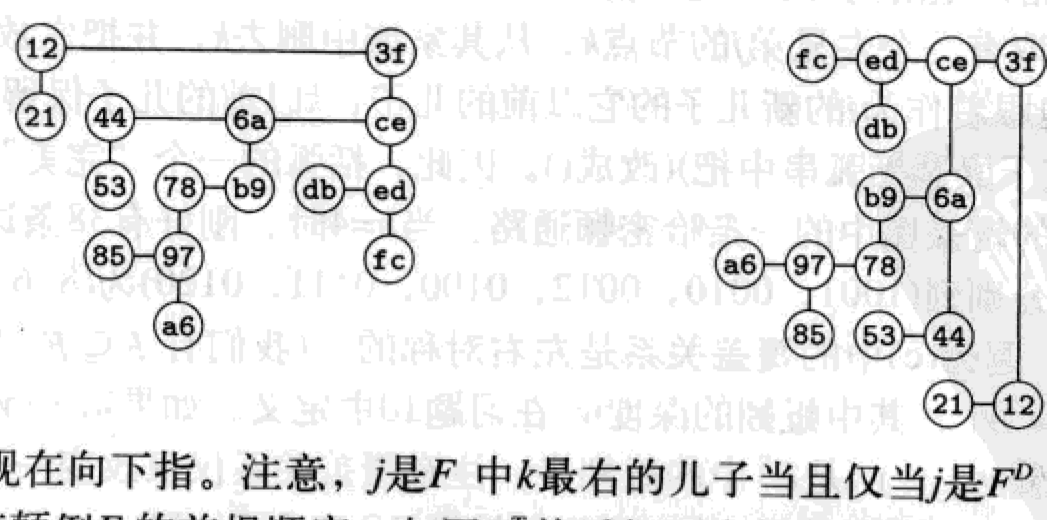
(b) 如果Π和Π'满足不交叉条件，则它们的最大公共求精 $\Pi \vee \Pi'$也是，所以，我们可以像在习题7.2.1.5-12(a)中那样进行。

(c) 令$x_1, \cdots, x_m$是F中某个节点的儿子，且令$1 \leqslant j < k \leqslant m$。通过从它们的家族中删除$x_{j+1}, \cdots, x_k$，把它们再附加为$x_{j+1}-1$的儿子，即$x_j$的最右后裔，来形成$F'$。

(d) 由(c)，这很显然。因此，这些森林自底向上通过它们所包含的非叶节点数来排阶(它比Π中块的个数少1)。

(e) 根的个数恰为 $\sum_{k=0}^{n} e_k(e_k-1)/2$，其中$e_0=n-e_1-\cdots-e_n$。

(f) 对偶性类似于习题12中转置操作，但我们使用左兄弟右儿子链接代替左儿子和右兄弟，关于小对角线进行转置：



(“右”链接现在向下指。注意，j是F中k最右的儿子当且仅当j是F^D中k的左兄弟。F^D的前根顺序颠倒F的前根顺序，如同F^T的后根顺序颠倒F的后根顺序那样。)

(g)由(f)我们可以看出F'覆盖F当且仅当F^D覆盖F'^D。(因此，如果F有k，则F^D有$n+1-k$个叶。)

(h) $F \wedge F' = (F^D \vee F'^D)^D$。

(i)否。如果它成立，则由对偶性，相等也必然成立。但是，例如，0101 $\wedge$

0121=0000，且0101 $\veebar$ 0121=0123，而叶(0101)+叶(0121)≠叶(0000)+叶(0123)。

[非交叉分划首先是由哈·贝克在*Math. Mag.* **22**(1948)，23-26中考虑的。格·克利维拉斯于1971年证明，它们形成一个格。参见答案2.3.4. 6-3中的参考

81 文献]。

27. (a) 这个断言等价于习题2. 3. 3-19。

(b) 如果我们通过右儿子和左兄弟链接表示一个森林，则前根顺序对应于二叉树的中根顺序(参见习题2.3.2-5)，而且s_j是节点j的右子树的大小。在这个二叉树的任何非叶节点向左转动恰减少范围坐标之一，而且减少的数量尽可能小，同一个有效表$s_1 \cdots s_n$相一致。因此F'覆盖F当且仅当F由F'通过这样的转动得到。(在左儿子/右兄弟的表示中的转动是类似的，但相对于后根顺序)。

(c) 对偶维持覆盖关系但交换左和右。

(d) $F \top F' = (F^D \perp F'^D)^D$。等价地，如习题6.2.3-32中所说明的，我们可独立地极小化左子树的大小。

(e) 显然，对于所有j，答案26(c)中的覆盖转换使$s_j \leqslant s'_j$。

(f) 真的。因为$F \barwedge F' \ltimes F \dashv F \perp F'$和$F \barwedge F' \ltimes F' \dashv F \perp F'$。

(g) 假的。例如0121 $\veebar$ 0122=0123，0121 $\top$ 0122=0122。(但通过(f)中取对偶，我们确实有$F \top F' \dashv F \veebar F'$。

(h) 长度$\binom{n}{2}$的最长通路重复地使最右非0的s_j减1。长度为$n-1$的最短通路重复地置最左非零的s_j为0。

答案6. 2. 3-32给出对于塔马利格的参考文献的许多引用。

28. (a) 只计算$\min(c_1,c'_1)\cdots\min(c_n,c'_n)$以及$\max(c_1,c'_1)\cdots\max(c_n,c'_n)$，因为$c_1 \cdots c_n$是一个有效序列当且仅当$c_1=0$，而且对于$1<j\leqslant n$，$c_j \leqslant c_{j-1}+1$。

(b) 由于(a)，显然如此。注：任何分配格的元素都可表示为某个偏序的序理想。在图41的情况下，该偏序图示于右边，而有长度为$n-2$的边的类似三角形栅格产生阶为n的斯坦利格。



(c) 从F中取有一个左兄弟j的节点k。从其家族中删去k，并把它放置成j的一个新右儿子，后边跟着作为j的新儿子的它以前的儿子；k以前的儿子保留自己的后裔。(这个操作对应于嵌套括弧串中把)(改成()。因此，括弧的一个“完美”格雷码对应于在斯坦利格的覆盖图中的一条哈密顿通路。当n=4时，刚好有38条这样的通路存在，即从0123分别到(1001，0010，0012，0100，0111，0120)为(8, 6, 6, 8, 4, 6。)

(d) 真的。因为(c)中的覆盖关系是左右对称的。(我们有$F \subseteq F'$当且仅当对于$0 \leqslant j \leqslant 2n$，$w_j \leqslant w'_j$，其中蚯蚓的深度$w_j$在习题10中定义。如果$w_0 \cdots w_{2n}$是$F$的蚯蚓步，则它的颠倒$w_{2n} \cdots w_0$是$F^R$中的蚯蚓步。注意覆盖关系仅仅改变一个坐标$w_j$。通过取诸$w$的min和max代替取诸$c$的，我们可计算$F \cap F'$和$F \cup F'$。

(e) 参见习题9。(因此，如习题27(f)，$F \perp F' \subseteq F \cap F'$等。)

注：斯坦利在*Fibonacci Quarterly* **13**(1975)，222-223中介绍了这个格。由于对相同的元素定义了三个重要的格，对于不同的顺序我们需要三个记号，这里采用的符号$\ltimes$, $\dashv$和$\subseteq$，旨在纪念克利维拉斯·塔马利和斯坦利(在俄语中，他是Стенли。

29. 如果把6个正五边形粘在一起，我们得到14个顶点，其坐标经过适当的转动和缩放之后分别为

$$p_{1010} = p_{0000}^{-} = p_{3000}^{*} = p_{2100}^{*-} = (-1, \sqrt{3}, 2/\phi)$$
$$p_{0010} = p_{3100}^{*} = (\phi^{-2}, \sqrt{3}\,\phi, 0); \quad p_{3010} = p_{0100}^{-} = (0,0,2); \quad p_{3210} = p_{0200}^{-} = (2,0,2/\phi)$$
$$p_{0210} = p_{3200}^{*} = (\sqrt{5}, \sqrt{3}, 0); \quad p_{1000} = p_{2000}^{*} = (-\phi^{2}, \sqrt{3}/\phi, 0)$$
82

$(x, y, z)^*$意味着$(x, -y, z)$而$(x, y, z)^-$意味着$(x, y, -z)$。但是，三个四边的“面”并非正方形。事实上，它们甚至不在一个平面上。

(然而，人们能通过把两个适当的四面体粘在一起，而且剪去三个粘在一起的角，得到看起来类似的一个固体，而且是真的正方形而非不规则的五边形。冈特·齐格勒在*Lectures on Polytopes* (纽约：Springer，1995)例9.11中讨论了数学上相当有兴趣，但不太吸引眼球的结合多边形的坐标的另一些可能集合)。

30. (a) $\bar{f}_{n-1} \cdots \bar{f}_1 0$，因为按对称顺序内节点$j$有一棵非空右子树当且仅当按对称顺序内节点$j+1$有空的左子树。

(b) 一般，如果足迹是$1^{p_1}0^{q_1+1}1^{p_2+1}0^{q_2+1}\cdots1^{p_k+1}0^{q_k+1}$，则我们要统计在对称顺序下其节点有描述$R^{p_1}NL^{q_1}BR^{p_2}NL^{q_2}B\cdots R^{p_k}NL^{q_k}$的所有二叉树，其中$B$指“两个子树都非空”，$R$指“右子树非空但左子树不是”，而$L$指“左子树为非空但右子树不是”，$N$指“没有子树非空”。这个数一般为

$$\binom{p_1+q_1}{p_1}\binom{p_2+q_2}{p_2}\cdots\binom{p_k+q_k}{p_k}C_{k-1}$$

特别地，它是$\binom{1+0}{1}\binom{0+0}{0}\binom{1+0}{1}\binom{5+3}{5}\binom{0+0}{0}\binom{0+0}{0}\binom{0+2}{0}\binom{0+0}{0}\binom{1+2}{1}C_8 = 240240$。

(c) 由习题3，$d_j=0$当且仅当$c_{j+1}>c_j$。

(d) 一般，由习题27(a) $F \perp F'$的足迹是$f_1\cdots f_n \wedge f_1'\cdots f_n'$；由(a)和习题27(d)，$F \top F'$的足迹是$f_1\cdots f_n \vee f_1'\cdots f_n'$。

[在塔马利格中补总存在，归功于哈·拉克塞尔。参见乔·格拉特泽尔，*General Lattice Theory* (1978)，习题I.6.30。]

31. (a)2^{n-1}。参见习题6. 2. 2-5。

(b) $c_1 \leqslant \cdots \leqslant c_n$；$d_1, \cdots, d_n \leqslant 1$。对于某个$j$，$e_j>0$意味着$e_j+\cdots+e_n=n-j$，$k_{j+1} \leqslant k_j+1$；$p_1 \leqslant \cdots \leqslant p_j \geqslant \cdots \geqslant p_n$；$s_j>0$意味着$s_j=n-j$；$u_1 \geqslant \cdots \geqslant u_n$；$z_{j+1} \leqslant z_j+2$。(一般也适用其他限制，在每种情况下削减可能性的数目到2^{n-1}。例如，$u_1 \cdots u_n$必须是范围坐标的一个有效序列。)

(c) 仅在2^{n-1}中的n种情况为真。(但F^T是退化的。)

(d) 有足迹$f_1 \cdots f_n$的退化森林有$c_{j+1}=c_j+f_j$。元素$j<k$是兄弟当且仅当$f_j=f_{j+1}=\cdots=f_{k-1}=0$。如果$F''$是具有足迹$f_1\cdots f_n \wedge f_1'\cdots f_n'$的退化森林，则$F'' \ltimes F$和$F'' \ltimes F'$。因此，$F'' \ltimes F \barwedge F' \dashv F \perp F'$。由(b)，我们也有$F \perp F' \dashv F''$。一个类似的论断证明$F \veebar F' = F \top F'$是具有足迹$f_1\cdots f_n \vee f_1'\cdots f_n'$的退化森林。

这样，当把克利维拉斯和塔马利格限制为退化森林时，它们就等于{1, ⋯, $n-$

1}子集的布尔格。[这个结果在塔马利格的情况下，由乔治·马尔科沃斯基给出，*Order* **9**(1992)，265-290，他的文章也证明塔马利格具有许多进一步的性质。]

32. 假设F和F'有范围坐标$s_1 \cdots s_n$和$s'_1 \cdots s'_n$。如果$s_j < s'_j$或$j=0$，则说下标j冻结。我们要描述冻结坐标的值并且极大化其他值。令$s_0=n$，对于$0 \leqslant k \leqslant n$，令

$$s''_k = s_j - k + j，其中j=\max\{i \mid 0 \leqslant i \leqslant k，i是冻结的，i+s_i \geqslant k\}。$$

83 由于当$0 \leqslant k-j \leqslant s_j$时$s_k \leqslant s_j-(k-j)$，我们有$s''_k \geqslant s_k$，当$k$冻结时相等。

按照习题27(a)的条件，范围坐标$s''_0 s''_1 \cdots s''_n$对应于一个有效的森林。如果$k \geqslant 0$，$0 \leqslant l < s''_k = s_j - k + j$，$s''_{k+l} = s_{j'} - k - l + j'$，如果$0 \leqslant j'-j \leqslant s_j$我们有$s''_{k+l} + l \leqslant s''_k$，因为这时，$s_{j'} + j' - j \leqslant s_j$。而且不能有$j>j'$或$j'>j+s_j$，因为$j+s_j \geqslant k+l \geqslant j'$。

令F''为范围满足$s_k \leqslant s'''_k \leqslant s''_k$的森林，则$\min(s'_k, s'''_k)=s_k$，因为当$k$冻结时，$s_k = s''_k$。否则$s_k = s'_k$。

反过来，如果F'''是满足$F' \perp F''' = F$的森林，必有$s_k \leqslant s'''_k \leqslant s''_k$。因为$s'''_k < s_k$意味着$s'''_k < s'_k$。如果$k$是满足$s'''_k > s''_k$的极小，对于某个冻结的$j$和$0 \leqslant j \leqslant k$和$j+s_j \geqslant k$，我们有$s''_k = s_j - k + j$。那么，$s'''_j \geqslant s_j$意味着$k-j \leqslant s'''_j$，因此$s'''_k + k - j \leqslant s'''_j$。如果$j<k$，我们有$s'''_j \leqslant s''_j = s_j$，矛盾。但$j=k$意味着$\min(s'''_k, s'_k) > s_k$。

为获得第一个半分配律，以$F \perp G$代替F和以F代替F'应用这个原理；然后，假设$F \dashv G \dashv F''$和$F \dashv H \dashv F''$意味着$F \dashv G \top H \dashv F''$。第二半分配律由第一个取对偶得出。

(拉尔芙·弗里西建议把F''称为F'对于F的虚拟补。)

33. (a)如果LLINK[k]$\neq 0$令$k\lambda$=LLINK[k]；否则，如果$k \neq 1$为RLINK[$k-1$]；否则，为二叉树的根。这个规则定义一个排列，因为$k\lambda=j$当且仅当k=parent(j)+[j是一个右儿子]，或者k=1而且j是根。而且当LLINK[k]=0时$k\lambda \geqslant k$以及当RLINK[k]=0时$k\sigma\lambda \leqslant k$。[对于推广到$t$叉树的情形，参见鲍·亨·埃德尔曼，*Discrete Math.* **40**(1982)，171-179。]

(b) 使用在答案26(f)中(2)的表示，我们看到在该情况下$\lambda(F)$是(3 1)(2)(12 6 4)(5)(11 7)(14 13)(9 8)(15)(10)。一般，这些循环是在每一循环中在递减顺序森林的家族；节点在前根顺序下编号。[参见德斯霍维兹和扎克斯，*Discrete Math.* **62**(1986)，215-218。]

(c) $\lambda(F^D)=\rho\sigma\lambda\rho$，其中$\rho$是“触发”排列$(1\ n)(2\ n-1)\cdots$，因为对偶森林交换LLINK↔RLINK和触发前根顺序的编号。

(d) 循环分裂$(x_j\ x_k)(x_1 \cdots x_m)=(x_1 \cdots x_j\ x_{k+1} \cdots x_m)(x_{j+1} \cdots x_k)$对应于答案26(c)。

(e) 由(d)，每个覆盖通路对应于$(n \cdots 21)$的一个因式分解。令q_n表示这种因式分解的个数。于是有递归式$q_1=1$和$q_n = \sum_{l=1}^{n-1}(n-l)\binom{n-2}{l-1} q_l\, q_{n-l}$。因为对于$k-j=l$，有$n-l$个选择，通过它第一个转置把循环分解成大小为$l$和$n-l$的一些部分。那么有$\binom{n-2}{l-1}$种方法来穿插相继的因式。这个解为$q_n=n^{n-2}$，因为

$$\sum_{l=1}^{n-1}\binom{n-1}{l}l^{l-1}(y-l)^{n-1-l}=\lim_{x\to 0}\sum_{l=1}^{n-1}\binom{n-1}{l}(x+l)^{l-1}(y-l)^{n-1-l}$$
$$=\lim_{x\to 0}\frac{(x+y)^{n-1}-y^{n-1}}{x}=(n-1)y^{n-1}$$

[参见约 · 迪尼斯，*Magyar Tudományos Akadémia Matematikai Kutató Intézetének Közleményei* **4**(1959)，63-70。寻求因式分解和标号自由树之间的一个对应是自然的，因为后者刚好也有n^{n-2}个。也许最简单的方法如下，给定$(1\ 2\cdots n)=(x_1\ y_1)\cdots(x_{n-1}\ y_{n-1})$，其中$x_j<y_j$。假设在$(x_j\ y_j)\cdots(x_{n-1}\ y_{n-1})$中包含$x_j$ 和y_j 的循环是$(z_1\ \cdots\ z_m)$，其中$z_1<\cdots<z_m$。如果$y_j=z_m$，令$a_j=z_1$，否则令$a_j=\min\{z_i\,|z_i>x_j\}$。于是人们能够证明，$a_1\cdots a_{n-1}$是停放$n-1$辆车的一个"叫醒序列"，而且习题6. 4-31把它同自由树联系起来。] 84

34. 从底到顶的每一个覆盖通路等价于形状为$(n-1, n-2, \cdots, 1)$的一个杨氏表景，所以，我们能使用定理5.1.4H(参见习题5. 3. 4-38)。

[在塔马利格中对于这样通路的枚举仍保持神秘，相关的序列为1, 1, 2, 9, 98, 2981, 340549, …]。

35. 乘以$n+1$，然后参看*AMM* **97**(1990)，626-630。

36. 对任意的$t\geqslant 1$，我们也可以推广到t叉树，办法是通过对步骤T1～T5做明显的修改。令$C_n^{(t)}$为具有n个内部节点的t叉树的数目；于是$C_n=C_n^{(2)}$和$C_n^{(t)}=((t-1)n+1)^{-1}\binom{tn}{n}$。如果度数$b_j$ 的h在访问之间改变，则在$C_{n-x}^{(t)}$种情况下我们有$h\geqslant x$。所以，容易情况以$1-C_{n-1}^{(t)}/C_n^{(t)}\approx 1-(t-1)^{t-1}/t^t$ 的概率出现。而且在步骤T4中$b_j\leftarrow 0$的平均次数为$(C_{n-1}^{(t)}+\cdots+C_1^{(t)})/C_n^{(t)}\approx(t-1)^{t-1}/(t^t-(t-1)^{t-1})$，或当$t=3$时为4/23。

确实，当$0\leqslant(t-1)p\leqslant q\neq 0$时我们也能研究$t$叉递归结构 $A_{pq}^{(t)}=0\ A_{p(q-1)}^{(t)}, tA_{(p-1)q}^{(t)}$，并且推广(5)。这样度数序列的个数 $C_{pq}^{(t)}$，满足递归式(21)，仅当$p<0$或$(t-1)p>q$时例外，$C_{pq}^{(t)}=0$。通解是

$$C_{pq}^{(t)}=\frac{q-(t-1)p+1}{q+1}\binom{p+q}{p}=\binom{p+q}{p}-(t-1)\binom{p+q}{p-1}$$

而且我们有 $C_n^{(t)}=C_{n((t-1)n)}^{(t)}$。对于$t=3$的三角形如右边所示那样。

1				
1				
1	1			
1	2			
1	3	3		
1	4	7		
1	5	12	12	
1	6	18	30	
1	7	25	55	55
1	8	33	88	143

37. 对所有这样的森林的基本词典顺序递归是，当$n_0>n_2+2n_3+\cdots+(t-1)\,n_t$ 和$n_1, \cdots, n_t\geqslant 0$时

$$\begin{aligned}A(n_0,n_1,\ldots,n_t) \;=\; & 0\,A(n_0-1,n_1,\ldots,n_t),\\ & 1\,A(n_0,n_1-1,\ldots,n_t),\ \ldots,\ t\,A(n_0,n_1,\ldots,n_t-1)\end{aligned}$$

否则，$A(n_0, n_1,\cdots, n_t)$为空，除非$A(0, 0, \cdots, 0)=\varepsilon$是由空串组成的序列。步骤G1计算$A(n_0, \cdots, n_t)$的第一项。我们要分析五个量：

C，G2执行的次数(森林的总数)；

E，从G3转向G2的次数(容易情况的个数)；

K，G4把某个b_i移到表c中的次数；

L，G5把b_j同某个c_i作比较的次数；

Z，G5置$c_1\leftarrow 0$的次数。

然后，在步骤G6中的循环置$b_{l-c_j}\leftarrow c_j$总共$K-Z-n_1-\cdots-n_t$次。

令n为向量$(n_0, n_1, \cdots, n_t)$，并令e_j为在坐标位置j处有1的单位向量。令$|n|=n_0+n_1+\cdots+n_t$和$\|n\|=n_1+2n_2+\cdots+tn_t$。使用这个记号，我们可以以下列方便的形式重写上边基本的递归式，当$|n|>\|n\|$时，

$$A(n)=0\ A(n-e_0),\ 1\ A(n-e_1),\ \cdots,\ t\ A(n-e_t)$$

考虑一般的递归关系

85
$$F(n)=f(n)+\left(\sum_{j=0}^{t} F(n-e_j)\right)\left[|n|>\|n\|\right],$$

而且每当向量n有一个负的分量时，$F(n)=0$。如果$f(n)=[|n|=0]$，则$F(n)=C(n)$是森林的总数。答案2. 3. 4. 4－32告诉我们

$$C(n)=\frac{(|n|-1)!(|n|-\|n\|)}{n_0!n_1!\cdots n_t!}=\sum_{j=0}^{t}(1-j)\binom{|n|-1}{n_0,\cdots,n_{j-1},n_j-1,n_{j+1},\cdots,n_t}$$

并且推广答案26中对于$C_{pq}^{(t)}$的公式(它是$n_0=(t-1)q+1$和$n_t=p$的情况)。类似地，通过选择其他的核心函数$f(n)$，我们得到在我们的分析中所需要的其他量$E(n)$，$K(n)$，$L(n)$和$Z(n)$的递归式：

$f(n)=[|n|=n_0+1$和$n_0>\|n\|]$	产生	$F(n)=E(n)$；
$f(n)=[|n|>n_0]$	产生	$F(n)=E(n)+K(n)$；
$f(n)=[|n|=\|n\|+1]$	产生	$F(n)=C(n)+K(n)-Z(n)$；
$f(n)=\sum_{1\leqslant j<k\leqslant t} n_j\,[n_k>0]$	产生	$F(n)=L(n)$。

习题2. 3. 4. 4-32的符号方法似乎产生不出对于这些更一般的递归式的快速解，但通过注意在步骤G2中$b_m+m<N$当且仅当前一步骤是G3，我们可以容易地建立$C-E$的值。因此，

$$C(n)-E(n)=\sum_{j=1}^{t} C(n-f_j)\text{ ，其中 } f_j=e_j-(j-1)\,e_0;$$

这个和统计出子森林，其中$n_1+\cdots+n_t$，即内部(非叶)节点的个数已减1。类似地，我们可以令

$$C^{(x)}(n)=\sum\{C(n-i_1f_1-\cdots-i_tf_t)\mid i_1+\cdots+i_t=x\}$$

是有$n_1+\cdots+n_t-x$个内部节点的子森林数。那么，我们有

$$K(n)-Z(n)=\sum_{x=1}^{|n|} C^{(x)}(n)$$

这是一个类似于(20)的公式，因为在步骤G5中$k-[b_j=0]\geqslant x\geqslant 1$当且仅当$b_{m-x}>0$和

$b_{m-x+1} \geqslant \cdots \geqslant b_m$。这样的前根顺序度的串是一对一地对应于 $C_{(n)}^{(x)}$的森林，如果我们从串$b_1 \cdots b_N$删去$b_{m-x+1} \cdots b_m$和适当个数的尾部0的话。

由这些公式我们可以得出结论，扎克斯－理查德算法每当$n_1=n_2+\cdots+n_t+O(1)$时，对于被访问的每个森林仅须$O(1)$个操作，因为当$j>1$时，$C(n-f_j)/C(n)=n_j n_0^{\underline{j-1}}/(|n|-1)^{\underline{j}} \leqslant 1/4+O(|n|^{-1})$。其实，在近乎所有实际感兴趣的情况下，$K$的值十分小。然而，当$n_1$很大时，算法可能很缓慢。例如，如果$t=1$，$n_0=m+r+1$，以及$n_1=m$，则这个算法实际上计算从$m+r$个事物取$r$个的所有组合；则 $C(n)=\binom{m+r}{r}$，而且当r固定时，$K(n)-Z(n)=\binom{m+r}{r+1}=\Omega(mC(n))$。[为确保在所有情况下的效率，我们可记住尾部的1；参见拉斯基和鲁兰特范·巴罗纳伊吉恩，*Congressus Numerantium* **41**(1984)，53-62]。

K，Z和(特别是)L的精确公式并不显得简单，但我们可以计算这些量如下：说一个森林的“活动块”是非零度的最右子串。例如，302102021230000000的“活动块”是2123。活动块的所有排列同等频繁地出现。其实，令$D(n)$表示对于说明n的森林的所有前根顺序度的串β的“尾部0$(\beta)-1$”之和。对于$1\leqslant j\leqslant t$，具有j的 n_j' 86
次出现的一个块精确地活动在 $D(n-n_1'f_1-\cdots-n_t'f_t)+[n_1'+\cdots+n_t'=n_1+\cdots+n_t]$ 情况下。例如，给定串3021020000，我们可以在3个位置插入21230000，以得到有活动块2123的一个森林。当这个活动块被刷出(即不以任何0居前)时，K和L的贡献可以像在习题7.2.1.2-6那样计算出来，即在该答案的记号下

$$k(n)=w(e_{n_1}(z)\cdots e_{n_t}(z)),\quad l(n)=w\left(e_{n_1}(z)\cdots e_{n_t}(z)\sum_{1\leqslant i\leqslant j\leqslant t}(n_i-zr_i(z))r_j(z)\right)$$

类似的贡献一般地出现；因此，对所有向量n'求和，使得对于$1\leqslant j\leqslant$ t，$n_j'\leqslant n_j$，而且$|n'|-\|n'\|=|n|-\|n\|$和$n_1'+\cdots+n_t'\leqslant n_1+\cdots+n_t-2$，

$$K(n)=k(n)+\sum D(n-n')k(n'),\ L(n)=l(n)+\sum D(n-n')\,l(n'),\ Z(n)=\sum D(n-n')$$

接下来确定$D(n)$。令$C(n;j)$是前根顺序最后的内节点有度数j的说明$n=(n_0,\cdots,n_t)$的森林个数，于是，我们有

$$C(n)=\sum_{j=1}^{t}C(n;j)\text{ 和 }C(n+e_1;1)=C(n+e_2;2)=\cdots=C(n+e_t;t)=C(n)+D(n)$$

由这个无穷的线性方程组，我们可以推导出$C(n)+D(n)$是

$$\sum_{i_2=0}^{n_2}\cdots\sum_{i_t=0}^{n_t}(-1)^{i_2+\cdots+i_t}\binom{i_2+\cdots+i_t}{i_2,\cdots,i_t}C(n+(1+i_2+\cdots+i_t)e_1-i_2f_2-\cdots-i_tf_t).$$

如果更简单的表达式存在，则当然它们是受欢迎的。

38. 步骤L1显然使用$4n+2$次内存访问。步骤L3恰好C_j-C_{j-1}次以一个特定的j值出到L4或L5，因此它花费$2C_n+3\sum_{j=0}^{n}(n-j)(C_j-C_{j-1})=2C_n+3(C_{n-1}+\cdots+C_1+C_0)$次内

存访问。步骤L4和L5一起共花费$6C_n-6$。因此整个过程对每个访问要有$9+O(n^{-1/2})$次内存访问。

39. 形状为(q, p)的一个杨氏表景和A_{pq}的一个元素对应于项y_{ij}，A_{pq}在位置$p+q+1-y_{21}, \cdots, p+q+1-y_{2p}$有左双亲，并在位置$p+q+1-y_{11}, \cdots, p+q+1-y_{1q}$有右双亲。钩的长度为$\{q+1, q, \cdots, 1, p, p-1, \cdots, 1\}\backslash\{q-p+1\}$。所以，由定理5.1.4H，$C_{pq}=(p+q)!(q-p+1)/(p!(q+1)!)$。

40. (a) $C_{pq}=\binom{p+q}{p}-\binom{p+q}{p-1}\equiv\binom{p+q}{p}+\binom{p+q}{p-1}=\binom{p+q+1}{p}$ (modulo 2)；现在使用习题1.2.6-11。(b) 由等式7.1-(oo)，我们知道$\nu(n\&(n+1))=\nu(n+1)-1$。

41. 它等于$C(wz)/(1-zC(wz))=1/(1-z-wzC(wz))=(1-wC(wz))/(1-w-z)$，其中$C(z)$是卡塔兰生成函数(18)。容易看出，这些公式的第一个$C(wz)+zC(wz)^2+z^2C(wz)^3+\cdots$等于(24)。[参见珀·阿·麦克马洪，*Combinatory Analysis* **1**(剑桥大学出版社，1915)，128-130。]

42. (a)元素$a_1\cdots a_n$确定一个完全自共轭嵌套串$a_1\cdots a_{2n}$，而且对于有q个右括弧的$a_1\cdots a_n$，恰好有$C_{q(n-q)}$个可能性。所以答案为

$$\sum_{q=0}^{\lfloor n/2\rfloor}C_{q(n-q)}=\sum_{q=0}^{\lfloor n/2\rfloor}\left(\binom{n}{q}-\binom{n}{q-1}\right)=\binom{n}{\lfloor n/2\rfloor}$$

(b) 精确地为$C_{(n-1)/2}[n为奇]$，因为一棵自转置二叉树由它的左子树确定。而且
87 (c)有相同的答案，因为F自对偶当且仅当F^R是自转置的。

43. 由对$q-p$的归纳法，有$C_{pq}=C_q-\binom{q-p-1}{1}C_{q-1}+\cdots=\sum_{r=0}^{q-p}(-1)^r\binom{q-p-r}{r}C_{q-r}$。

44. 在步骤B3中在访问之间访问内存的次数为$3j-2$；在步骤B4中为$h+1$；而在步骤B5中为4，其中h是$y\leftarrow r_y$的次数。给定j和x，当$j<n$时具有$h\geqslant x$的二叉树个数为$[z^{n-j-x-1}]C(z)^{x+3}$，因为通过把$x+3$个子树附加到$j+x+1$个内部节点之下，我们可以得到这样的树。置$x=0$告诉我们，使用(24)和习题43，一个给定的j值出现$C_{(n-j-1)(n-j+1)}=C_{n+1-j}-C_{n-j}$次。因此，对于所有的二叉树的$\sum j$是$n+\sum_{j=1}^{n}(C_{n+1-j}-C_{n-j})j=C_n+C_{n-1}+\cdots+C_1$。类似地，$\sum(h+1)$是$\sum_{j=1}^{n}\sum_{x=0}^{n-j-1}C_{(n-j-x-1)(n-j+1)}=\sum_{j=1}^{n-1}C_{(n-j-1)(n-j+2)}=\sum_{j=1}^{n}(C_{n-j+2}-2C_{n-j+1})=C_{n+1}-(C_n+C_{n-1}+\cdots+C_0)$。所以整个地，这个算法花费$C_{n+1}+4C_n+2(C_{n-1}+\cdots+C_1)+O(n)=(26/3-10/(3n)+O(n^{-2}))C_n$次内存访问。

45. 在步骤K3中的每个容易情况出现C_{n-1}次，因此该步骤的总花费是$3C_{n-1}+8C_{n-1}+2(C_n-2C_{n-1})$次内存访问。步骤K4取$r_i$总共$[z^{n-i-1}]C(z)^{i+2}=C_{(n-i-1)n}$次；对于$i\geqslant 2$求和给出在该循环中总共有$C_{(n-3)(n+1)}=C_{n+1}-3C_n+C_{n-1}$次内存访问。步骤K5花费$6C_n-12C_{n-1}$。步骤K6有点复杂，但是当$n>2$时人们能够证明操作$r_j\leftarrow r_k$执行了$C_n-3C_{n-1}+1$次，同时操作$r_j\leftarrow 0$执行了$C_{n-1}-n+1$次。因此，总共访问内存的次数为$C_{n+1}+7C_n-9C_{n-1}+n+3=(8.75-9.375/n+O(n^{-2}))C_n$。

尽管这个总数比答案44中算法B的总数渐近地要差些，但n^{-1}的大的负系数意

味着算法B仅当$n \geqslant 58$时才真正取胜，而n不总是这么大。

46. (a) 从 ⓟⓠ 向左使面积增加$q-p$。

(b) 从 ⓝⓝ 向 ⓞⓞ 的一条通路向左的步骤对应于$a_1 \cdots a_{2n}$中的左括弧，而且在第k个这样的步中我们有$q-p=c_k$。

(c) 等价地，$C_{n+1}(x)=\sum_{k=0}^{n} x^k C_k(x) C_{n-k}(x)$。这个递归式成立，因为$n+1$节点的森林$F$由最左树的根连同一个$k$节点的森林$F_l$(该根的后裔)和一个$(n-k)$节点的森林$F_r$(剩下的树)组成，而且因为我们有

$$\text{内部通路长度}(F)=k+\text{内部通路长度}(F_l)+\text{内部通路长度}(F_r)$$

(d) $A_{p(p+r)}$的串有形式$\alpha_0)\alpha_1)\cdots\alpha_{r-1})\alpha_r$，其中每个$\alpha_j$适当地嵌套。这个串的面积是对$j$个$\alpha_j$的面积求和，加上$r-j$乘以$\alpha_j$中左双亲的个数。

注：由伦·卡尔里兹和约·里奥丹在*Duke Math. J.* **31**(1964)，371-388中介绍的多项式$C_{pq}(x)$，在部分(d)中的恒等式等价于他们的公式(10.12)。他们还证明了

$$C_{pq}(x)=\sum_r (-1)^r x^{r(r-1)-\binom{q-p}{2}} \binom{q-p-r}{r}_x C_{q-r}(x)$$

并且推广了习题43的结果。从部分(c)有无穷连分式$C(x,z)=1/(1-z/(1-xz/(1-x^2z/(1-\cdots))))$，乔·尼·沃特森证明了它等于$F(x,z)/F(x,z/x)$，其中

$$F(x,z)=\sum_{n=0}^{\infty}\frac{(-1)^n x^{n^2} z^n}{(1-x)(1-x^2)\cdots(1-x^n)}$$

参见*J. London Math. Soc.* **4**(1929)，39-48。在习题5.2.1-15，我们已经遇到过稍经伪装的相同的生成函数。 88

一个森林的内部通路长度是对应二叉树的“左通路长度”，即从根开始的通路上左分支数的所有内部节点之和。对于所有n节点的二叉树T求和，更一般的多项式

$$C_n(x,y)=\sum x^{\text{左通路长度}(T)} y^{\text{右通路长度}(T)}$$

似乎不遵守在这个习题中研究的像对于$C_{nn}(x)=C_n(x,1)$这样的简单加法递归式，但我们确实有$C_{n+1}(x,y)=\sum_k x^k C_k(x,y)\, y^{n-k} C_{n-k}(x,y)$。因此，超生成函数$C(x,y,z)=\sum_n C_n(x,y)z^n$满足函数方程$C(x,y,z)=1+zC(x,y,xz)C(x,y,yz)$。(在习题2.3.4.5-5中考虑$x=y$的情况)。

47. 对于$0 \leqslant p<n$，$C_n(x)=\sum_q x^{\binom{q-p}{2}} C_{pq}(x)C_{(n-q)(n-1-p)}(x)$

48. 设在习题46的记号下$\overline{C}(z)=C(-1,z)$，并设$\overline{C}(z)\overline{C}(-z)=F(z^2)$。则$\overline{C}(z)=1+zF(z^2)$和$\overline{C}(-z)=1-zF(z^2)$；所以，$F(z)=1-zF(z)^2$；而且$F(z)=C(-z)$。由此得出，$C_{pq}(-1)=[z^p]C(-z^2)^{\lceil (q-p)/2\rceil}(1+zC(-z^2))^{[q-p\text{ even}]}$，当$q$为偶数时，它是$(-1)^{(p/2)}C_{(p/2)(q/2-1)}[p\text{偶数}]$，当$q$为奇数时，为$(-1)^{\lfloor p/2\rfloor}C_{\lfloor p/2\rfloor\lfloor q/2\rfloor}$。通过$A_{pq}$的串，一个完美的格雷码能够存在仅当$|C_{pq}(-1)|\leqslant 1$时，因为相关联的图是二部分图(参见

图41)；$|C_{pq}(-1)|$是两部分大小之间的差，因为每个完美的转置改变$c_1+\cdots+c_n$的值±1。

49. 对于n=15和N=10^6执行算法U，它是()(()())((((()()))((((())()))))。

50. 对算法U作下列改动：在步骤U1，再置$r\leftarrow0$。在步骤U3中，检测是否a_m=‘)’代替检测是否$N\leqslant c'$。在步骤U4中，置$r\leftarrow r+c'$代替$N\leftarrow N-c'$。而且在步骤U3和U4中省略对a_m的赋值。

在(1)中的串结果有阶3141592(谁知道？)。

51. 由定理7.2.1.3L，$N=\binom{\bar z_1}{n}+\binom{\bar z_2}{n-1}+\cdots+\binom{\bar z_n}{1}$；因此$\kappa_n N=\binom{\bar z_1}{n-1}+\binom{\bar z_2}{n-2}+\cdots+\binom{\bar z_n}{0}$，因为$\bar z_n\geqslant1$。现在注意，$N-\kappa_n N$是$z_1\,z_2\cdots z_n$的阶，因为(23)和习题50所致。(例如，设$z_1\cdots z_4$=1256，它在表1中阶为6。然后，$\bar z_1\cdots\bar z_4$=7632，$N$=60，以及$\kappa_4\,60$=54。注意$N$相当大，因为$\bar z_1=2n-1$；图27显示，当$N$较小时，$\kappa_n N$通常超过$N$。)

52. 尾部右括弧的个数和前导的左括弧个数有相同的分布，而且以‘$(^k$’开始的嵌套串序列是$(^k A_{(n-k)(n-1)}$。因此，$d_n=k$的概率是$C_{(n-k)(n-1)}/C_n$。我们发现，使用等式1.2.6-(25)

$$\sum_{k=0}^{n}\binom{k}{t}C_{(n-k)(n-1)}=\sum_{k=0}^{n}\left(\binom{2n-1-k}{n-1}-\binom{2n-1-k}{n}\right)\binom{k}{t}$$
$$=\binom{2n}{n+t}-\binom{2n}{n+t+1}=C_{(n-t)(n+t)}$$

由此得出，均值和方差分别等于$3n/(n+2)=3-6/(n+2)$和$2n(2n^2-n-1/((n+2)^2(n+3))$
88 $=4+O(n^{-1})$。[这个分布的矩首先由赖·克姆普在*Acta. Informatica* **35**(1998)，17-89定理9中计算的。注意$c_n=d_n-1$实际上有相同的特性。]

53. (a) 由习题52，$3n/(n+2)$。(b) 由习题6.2.2-7，H_n。(c) 由归纳法，$2-2^{-n}$。

(d) 在遇到一片叶之前，任何左或右分支的特殊的(但固定的)序列有相同的步骤分布。(换言之，有德威二进记号01101的一个节点出现的概率和00000出现的概
89 率相同。)因此，如果$X=k$的概率为p_k，在层k上的2^k个可能节点的每一个是外部节点的概率为p_k。因此，预期值$\sum_k 2^k p_k$是外部节点的预期个数，在所有三种情况下为n+1。(人们当然能直接验证这个结果，且在情况(a)中$p_k=C_{(n-k)(n-1)}/C_n$，在情况(b)中$p_k=\left[{n \atop k}\right]/n!$，以及在情况(c)中$p_k=2^{-k+[k=n]}$。)

注：在这三种情况中平均路径长度结果是$\Theta(\sqrt{n})$，$\Theta(\log n)$和$\Theta(n)$；因此，当碰上一片叶的预期时间更短时，它是更长的!原因是接近于根的无处不在的“孔”迫使其他通路变长。情况(a)当在答案36的记号下$p_k=C^{(t)}_{(n-k)((t-1)n-1)}/C^{(t)}_n$时，对于$t$叉树有一个有趣的推广。于是，到一片叶的均值距离是$(t+1)n/((t-1)n+2)$，通过望远镜序列来证明

$$\sum_k t^k C^{(t)}_{(n-k)((t-1)n-1)}=\binom{tn}{n}$$

是有启发的。

54. 对x求微分，我们有

$$C'(x,z)=zC'(x,z)C(x,xz)+zC(x,z)(C'(x,xz)+zC_{\imath}(x,xz))$$

其中$C_{\imath}(x,z)$表示$C(x,z)$关于z的导数。于是，$C'(1,z)=2zC'(1,z)C(z)+z^2C(z)C'(z)$；由于$C'(z)=C(z)^2+2zC(z)C'(z)$，对于$C'(1,z)$我们可以求解，得到$z^2C(z)^3/(1-2z)C(z))^2$。因此，$\sum(c_1+\cdots+c_n)=[z^n]C'(1,z)=2^{2n-1}-\frac{1}{2}(3n+1)C_n$，这同习题2.3.4.5-5一致。类似地，我们求出

$$\sum(c_1+\cdots c_n)^2=[z^n]C''(1,z)=\left(\frac{5n^2+19n+6}{6}\right)\binom{2n}{n}-\left(1+\frac{3n}{2}\right)4^n$$

因此，均值和方差分别为$\frac{1}{2}\sqrt{\pi}n^{3/2}+O(n)$和$\left(\frac{5}{6}-\frac{\pi}{4}\right)n^{3/2}+O(n)$。

55. 像在答案54中一样求微分，使用习题46(d)和习题5.2.1-14的公式，连同$[z^n]C(z)^r/(1-4z)=2^{2n+r}-\sum_{j=1}^{r}2^{r-j}\binom{2n+j}{n}$，产生

$$\begin{aligned}C'_{p(p+r)}(1)&=[z^p]\left((r+1)\frac{z^2C(z)^{r+3}}{1-4z}+\binom{r+1}{2}\frac{zC(z)^{r+2}}{\sqrt{1-4z}}\right)\\&=[z^p]\left((r+1)\frac{C(z)^{r+1}-2C(z)^r+C(z)^{r-1}}{1-4z}+\binom{r+1}{2}\frac{C(z)^{r+1}-C(z)^r}{\sqrt{1-4z}}\right)\\&=(r+1)\left(2^{2p+r-1}-\binom{2p+r+1}{p}-\sum_{j=1}^{r-1}2^{r-1-j}\binom{2p+j}{p}\right)+\binom{r+1}{2}\binom{2p+r}{p-1}\end{aligned}$$

56. 使用1.2.6-53(b)。[参见*BIT* **30**(1990)，67-68。]

57. 由1.2.6-(21)，$2S_0(a,b)=\binom{2a}{a}\binom{2b}{b}+\binom{2a+2b}{a+b}$。习题1.2.6-53告诉我们

$$\sum_{k=a-m}^{a}\binom{2a}{a-k}\binom{2b}{b-k}k=(m+1)(a+b-m)\binom{2a}{m+1}\binom{2b}{a+b-m}$$

因此 $2S_1(a,b)=\binom{2a}{a}\binom{2b}{b}\frac{ab}{a+b}$。由于 $b^2S_p(a,b)-S_{p+2}(a,b)=S_p(a,b-1)$，我们发现 $2S_2(a,b)=\binom{2a+2b}{a+b}\frac{ab}{2a+2b-1}$；$2S_3(a,b)=\binom{2a}{a}\binom{2b}{b}a^2b^2/(a+b)^2$。通过设置$a=m$，$b=n-m$，以及$C_{(x-k)(x+k)}=\binom{2x}{x-k}-\binom{2x}{x-k-1}$得到公式(30)。

类似地，w_{2m-1}的平均值是$\sum_{k\geq 0}(2k-1)C_{(m-k)(m+k-1)}C_{(n-m-k+1)(n-m+k)}$除以$C_n$，即

90 $$\frac{2S_3(m,n+1-m)-S_2(m,n+1-m)}{m(n+1-m)C_n}=\frac{m(n+1-m)}{n}\binom{2m}{m}\binom{2n+2-2m}{n+1-m}\Big/\binom{2n}{n}-1$$

[赖·克姆普，*BIT* **20**(1980)，157-163；赫·普罗丁格尔，*Soochow J. Math.* **9**(1983)，193～196。]

58. 对左子树有k个内部节点的情况求和，我们有

$$t_{lmn}=[l=m=n=0]+\sum_{k=0}^{m-1}C_k t_{(l-1)(m-k-1)(n-k-1)}+\sum_{k=m}^{n-1}C_{n-1-k}t_{(l-1)mk}$$

因此三元组生成函数 $t(v,w,z)=\sum_{l,m,n}t_{lmn}v^l w^m z^n$ 满足

$$t(v,w,z)=1+vwzC(wz)t(v,w,z)+vzC(z)t(v,w,z)\,;$$

而且对于$t(w,z)=\partial t(v,w,z)/\partial v|_{v=1}$ 的类似线性关系由此得出，因为$t(1,w,z)=\sum_{n=0}^{\infty}\sum_{m=0}^{n}C_n w^m z^n=(C(z)-wC(wz))/(1-w)$而且$zC(z)^2=C(z)-1$。代数操作现在生成

$$t(w,z)=\frac{C(z)+wC(wz)-(1+w)}{(1-w)^2 z}-\frac{2wC(z)C(wz)}{(1-w)^2}-\frac{C(z)-wC(wz)}{1-w}\ ,$$

因此我们得到公式 $t_{mn}=(m+1)C_{n+1}-2\sum_{k=0}^{m}(m-k)C_kC_{n-k}-C_n$ 。现在

$$\sum_{k=0}^{m-1}(k+1)C_kC_{n-1-k}=\frac{m}{2n}\binom{2m}{m}\binom{2n-2m}{n-m}$$

可以像习题56中那样证明，由此推出，对于$0\leqslant m\leqslant n$，

$$t_{mn}=2\binom{2m}{m}\binom{2n-2m}{n-m}\frac{(2m+1)(2n-2m+1)}{(n+1)(n+2)}-C_n\ ,$$

[彼·克斯承霍夫尔，*J. Combintorics, Information and System Sciences* **8**(1983)，44-60。对于更高的矩和推广，参见沃·约·古特雅尔，*Random Structures and Algorithms* **3**(1992)，361-374。阿·潘霍尔泽和赫·普罗丁格尔，*J. Statistical Planning and Inference* **101**(2002)，267-279。注意生成函数$t(u,w,z)$生成

$$t_{lmn}=\sum_k\binom{l}{k}C_{(m-k)(m-1)}C_{(n-m-l+k)(n-m-1)}$$

当$m\geqslant 1$时利用 $\sum_k\binom{k}{r}C_{(n-k)(m-1)}=C_{(n-r)(m+r)}$ 这一事实，我们得到公式$t_{mn}+C_n=\sum_k(k+1)C_{(m-k)(m-1)}C_{(n-m)(n-m+k+1)}$，这是(令人惊讶地)以封闭形式表达的和。]

59. $$T(w,z)=\frac{w(C(z)-C(wz))}{(1-w)}-wzC(z)C(wz)+zC(z)T(w,z)+wzC(wz)T(wz)$$
$$=\frac{w((C(z)+C(wz)-2)/z-(1+w)C(z)C(wz)-(1-w)(C(z)-C(wz)))}{(1-w)^2}$$

因此$T_{mn}=t_{mn}-\sum_{k=m}^{n}C_kC_{n-k}$。[是否存在组合的证明？]而且，对于$1\leqslant m\leqslant n$，

$$T_{mn}=\binom{2m}{m}\binom{2n+2-2m}{n+1-m}\frac{4m(n+1-m)+n+1}{2(n+1)(n+2)}-\frac{1}{2}C_{n+1}-C_n$$

60. (a)在协原子中它是右括弧的个数。(因此，它也是在关联的“蚯蚓步行”中使$w_{2k-1}<0$的k的个数)。

(b)为方便起见，令d(‘(’)=+1和d(‘)’)=－1。

A1. [初始化。] 置$i\leftarrow j\leftarrow 1$和$k\leftarrow 2n$。

A2. [完成了吗？] 如果$j>k$，则算法终止。否则置$a_j\leftarrow$‘(’，$j\leftarrow j+1$。 91

A3. [是原子吗？] 如果b_i = ‘)’，置$s\leftarrow -1$，$i\leftarrow i+1$，并转到A4。否则置$s\leftarrow 1$，$i\leftarrow i+1$，当$s>0$时，置$a_j\leftarrow b_i$，$j\leftarrow j+1$，$s\leftarrow s+d(b_i)$，$i\leftarrow i+1$，返回到A2。

A4. [协原子。] 置$s\leftarrow s+d(b_i)$。如果$s<0$，置$a_k\leftarrow b_i$，$k\leftarrow k-1$，$i\leftarrow i+1$，重复步骤A4。否则，置$a_k\leftarrow$‘)’，$k\leftarrow k-1$，$i\leftarrow i+1$，并返回A2。 ▮

(c) (1)的缺失11的逆是(( )))((( ))))))(( )((( ))(( )))(((。一般，通过把下标m定为最后右括弧算起第l个位置之前来求，匹配括弧的下标(u_0, v_0), …, (u_{s-1}, v_{s-1})使得$u_j\leqslant m<v_j$。

I1. [初始化。] 置$c\leftarrow j\leftarrow s\leftarrow 0$，$k\leftarrow m\leftarrow 2n$，而且$u_0\leftarrow 2n+1$。

I2. [从右至左扫描。] 如果a_k= ‘)’，则转到I3。如果a_k = ‘(’，则转到I4；如果k=0，则转到I5。

I3. [处理一个‘)’。] 置$r_j\leftarrow k$，$j\leftarrow j+1$，$c\leftarrow c+1$。如果$c=l$，则置$m\leftarrow k-1$，$s\leftarrow j$并且$u_s\leftarrow k$。然后，k减1，返回到I2。

I4. [处理一个‘(’。] (这时左括弧a_k匹配右括弧$a_{r_{j-1}}$)。置$j\leftarrow j-1$。如果$r_j>m$，则置$u_j\leftarrow k$，以及$v_j\leftarrow r_j$。然后，k减1，返回I2。

I5. [准备排列。] 置$i\leftarrow j\leftarrow 1$，$k\leftarrow 2n$，并且$c\leftarrow 0$。

I6. [排列。] 当$j\neq u_c$时，置$b_i\leftarrow a_j$，$i\leftarrow i+1$，$j\leftarrow j+1$。如果$c=s$则停止。否则，置$b_i\leftarrow$‘)’，$i\leftarrow i+1$，$j\leftarrow j+1$。当$k\neq v_c$时，置$b_i\leftarrow a_k$，$i\leftarrow i+1$，$k\leftarrow k-1$。然后，置$b_i\leftarrow$‘(’，$i\leftarrow i+1$，$k\leftarrow k-1$，$c\leftarrow c+1$，重复I6。 ▮

注：对于$0\leqslant l\leqslant n$，长度为$2n$的恰好有C_n个平衡的串具有缺失l这一事实，是由珀·阿·麦克马洪发现的*Philosophical Transactions* **209**(1909)，153-175，§20]。然后又为钟开莱和威·费勒重新发现[*Proc. Nat. Acad. Sci.* **35**(1949)，605-608]，他们使用了生成函数。随后，由小约·劳·霍德格斯(J. L. Hodges Jr.)发现了一个简单的组合说明*Biometrika* **42**(1955)，261-262]，他发现，如果$\beta_1\cdots\beta_r$有缺失$l>0$，如果$\beta_k=\alpha_k^R$是它最右的协原子，则平衡串$\beta_1\cdots\beta_{k-1}(\beta_{k+1}\cdots\beta_r)\,\alpha_k'^R$有缺失$l-1$(这个变换是可颠倒的)。本题中的有效映射类似于迈·戴·埃特金森和约-鲁·萨克的构造*Information Processing Letters* **41**(1992)，21-23]。

61. (a)令$c_j=1-b_j$，这样$c_j\leqslant 1$，$c_1+\cdots+c_N=f$，我们必须证明

$$c_1+c_2+\cdots+c_k<f\qquad\text{当且仅当}k<N$$

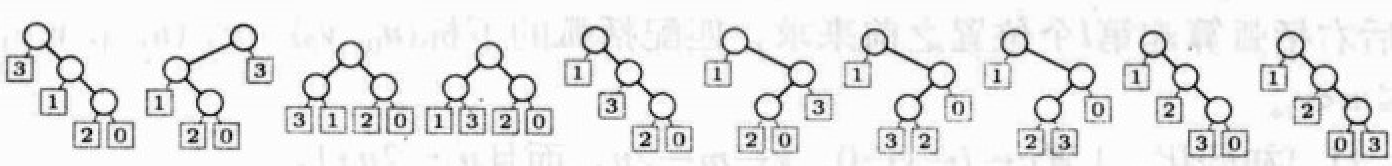
恰好对f个循环移动成立。通过令$c_{j\pm N}=c_j$，我们可以对所有整数j定义c_j。我们也通过令$\Sigma_0=0$和$\Sigma_j=\Sigma_{j-1}+c_j$，对所有$j$定义$\Sigma_j$。于是，$\Sigma_{j+Nt}=\Sigma_j+ft$，而且$\Sigma_{j+1}\leqslant\Sigma_j+1$。由此得出，对于每个整数$x$，存在一个最小的整数$j=j(x)$，使得$\Sigma_j=x$。而且，$j(x)<j(x+1)$以及$j(x+f)=j(x)+N$。因此，期望条件成立当且仅当对于$x=1, 2, \cdots$，或$f$，我们移动$j(x) \bmod N$位。(这个重要引理的历史在答案2. 3. 4. 4-32中讨论。)

(b) 以$l\leftarrow m\leftarrow s\leftarrow 0$开始。对于$k=1, 2, \cdots, N$(以这个顺序)作以下操作：置$s\leftarrow s+1-b_k$；如果$s>m$，则置$m\leftarrow s$，$j_l\leftarrow k$和$l\leftarrow(l+1) \bmod f$。由部分(a)的证明，答案为$j_0, \cdots, j_{f-1}$。

(c) 对于$0\leqslant j\leqslant t$，以包含j的n_j个出现的任何串$b_1 b_2\cdots\ b_N$开始。对这个串应用一个随机排列，然后应用部分(b)的算法。在$(j_0, \cdots, j_{f-1})$之间随机地选择，并且使用得到的循环移位作为前根顺序的序列来定义森林。

[关于更一般的算法，参见劳·阿伦索，吉·拉·勒密和仁·期库特，
92 *Algorithmica* **17**(1997)，162-182。]

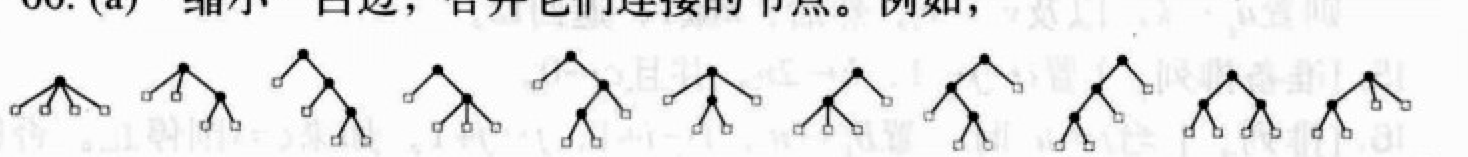
62. 二进位串$(l_1\cdots l_n,\ r_1\cdots r_n)$有效当且仅当习题20中的$b_1\cdots b_n$有效，其中$b_j=l_j+r_j$。因此，我们可以使用习题61。[参见詹·F·库斯，*Information Processing Letters* **45**(1993)，291-294。]

63. 

64. $X=2k+b$，其中$(k, b)=(0, 1)$，$(2, 1)$，$(0, 0)$，$(5, 1)$，$(6, 0)$，$(1, 1)$。终于$L_0 L_1\cdots L_{12}$=5 11 3 4 0 7 9 8 1 6 10 12 2。

65. 参见潘霍尔茨和赫·普罗丁格，*Discrete Mathematics* **250**(2002)，181-195；马·拉克扎克和彼·温克勒，*Random Structures and Algorithms* **24**(2004)，420-443。

66. (a)"缩小"白边，合并它们连接的节点。例如，



是对应于$n=3$所描述的11个施罗德树的通常树。在这个对应下，一个左链接意味着"这是一个儿子"。而白的右链接意味着"在此查找更多的儿子"。一个黑的右链接意味着，"这是最后的儿子"。

(b) 模拟算法L，但在转动之间使用一个通常的格雷二进制码来遍历无论存在什么样的右链接的所有颜色模式。(事实上，$n=3$的情况在例子中已加图解。)

注意如(53)中，施罗德树也对应于串-并行图。然而，它们确实对并行地连接的边和/或超边强加一个顺序；所以，它们像*在平面中的嵌套*那样，更精确地对应于串-并行图(除s和t之外，边和顶点都未加标号)。

67. $S(z)=1+zS(z)(1+2(S(z)-1))$。因为$1+2(S(z)-1)$枚举右子树，因此$S(z)=(1+z-\sqrt{1-6z+z^2}\,)/(4z)$。

注：在习题2.3.4.4-31中我们已经见过施罗德数，其中$G(z)=zS(z)$；在习题2.2.1-11中，对于$n\geqslant 2$，$b_n=2S_{n-1}$，而且我们还发现递归式$(n-1)S_n=(6n-3)S_{n-1}-(n-2)S_{n-2}$。如习题2.2.1-12中显露的那样，它们渐近地增长。数S_{pq}的一个三角形

类似于(22)，可用来生成随机施罗德树。这些数满足

$$\begin{aligned}S_{pq}&=S_{p(q-1)}+S_{(p-1)q}+S_{(p-2)q}+\cdots+S_{0q}=S_{p(q-1)}+2S_{(p-1)q}-S_{(p-1)(q-1)}\\&=\frac{q-p+1}{q+1}\sum_{k=0}^{p}\binom{q+1}{p-k}\binom{p-1}{k}2^k=\sum_{k=0}^{p}\left(\binom{q}{p-k}\binom{p-1}{k}-\binom{q}{p-k-1}\binom{p-1}{k-1}\right)2^k\\&=[w^pz^q]S(wz)/(1-zS(wz))\end{aligned}$$

最后一行的双重生成函数是由埃密里克 · 德特斯给出的。在理查德 · 斯坦利的 *Enumerative Combinatorics* **2**(1999)习题6. 39中讨论了施罗德树的许多其他性质。

68. 仅含空串ϵ的一行。(从$n-1$到n的一般规则(36)把这行转换成‘0 1’，即阶为1的模式。)

69. 如果我们忽略在每串开始处的‘10’，则前$\binom{6}{3}=20$行是阶数为6的圣诞树模式。阶为7的模式更难看出。但有$\binom{7}{3}=35$行，其中最左项以0开始，不理睬所有这些行最右的串，并且忽略在每个剩余串中开始处的0。(其他答案也是可能的。) 93

70. 如果σ出现在圣诞树模式的第k列，令σ'是同一行的$n-k$列中的串。(如果我们想的是括弧而不是二进位，由(39)，这个规则是在答案11的意义下自由括弧的镜面反射。)

71. M_{tn}是t个最大的二项式系数$\binom{n}{k}$之和，因为圣诞树模式的每行至多可包含S的t个元素，因为通过选择满足$(n-t)/2\leqslant\nu(\sigma)\leqslant(n+t-1)/2$的所有串$\sigma$，确实得到一个这样的集合$S$。(公式

$$M_{tn}=\sum_{n-t\leqslant 2k\leqslant n+t-1}\binom{n}{k}$$

大概是尽可能简单的。然而，对于小的t，像$M_{(2)n}=M_{n+1}$这样的特殊公式成立，而且对于$t>n$，我们也有$M_{tn}=2^n$。)

72. 你得到和在上题中相同的数M_{sn}。事实上，通过归纳法，人们能够证明恰好有$\binom{n}{n-k}-\binom{n}{k-s}$长度为$s+n-2k\geqslant 0$的行。

73. 011001001000000000100101001100，111001011011111111101101011100；参见(38)。

74. 通过词典顺序性质，我们要统计最右元素分别有形式$0*^{29}$，$10*^{28}$，$110*^{27}$，$111000*^{24}$，$11100100*^{22}$，$111001010*^{21}$，$11100101100*^{19}$，$111001011010*^{18}$，$1110010110110*^{17}$，…，即所有以τ=111001011011111111101101011100居前的所有30个二进位的串的行数。

如果θ中1的位数比0的位数多p个，则以$\theta*^n$结尾的圣诞树的行数和以1^p*^n结尾的行数相同。而且由习题71，这是$M_{(p+1)n}$，因为所有这样的行是起始行‘$0^p0^{p-1}1\cdots1^p$’

的n步后裔。

结果，答案为$M_{0(29)}+M_{1(28)}+M_{2(27)}+M_{1(24)}+\cdots+M_{(12)3}+M_{(13)2}=\sum_{k=1}^{21}M_{(2k-1-z_k)(n-z_k)}$ $=0+\binom{28}{14}+\binom{27}{14}+\binom{27}{13}+\binom{24}{12}+\cdots+8+4=84867708$，其中$(z_1,\cdots,z_{21})=(1,2,3,6,\cdots,27,28)$是1在$\tau$中出现的位置序列。

75. 我们有$r_1^{(n)}=M_{n-2}$，因为行$r_1^{(n)}$是(33)中第一行的底部后裔。由答案74中的公式，我们也有$r_{j+1}^{(n)}-r_j^{(n)}=M_{j(n-1-j)}-M_{(j-1)(n-2-j)}=M_{(j+1)(n-2-j)}$，因为对于行$r_j^{(n)}$的相关序列$z_1\cdots z_{n-1}$是$1^j0\,1^{n-1-j}$。因此，由于当$n\to\infty$时对于固定的$j$，$M_{jn}/M_n\to j$，我们有

$$\lim_{n\to\infty}\frac{r_j^{(n)}}{M_n}=\sum_{k=1}^{j}\frac{k}{2^{k+1}}=1-\frac{j+2}{2^{j+1}}$$

而且我们也含蓄地证明了$\sum_{k=0}^{n}M_{k(n-k)}=M_{n+1}-1$。

76. 无穷序列

Q=1313351313351335355713133513133513353557131335133535571335355735575779⋯

的前$\binom{2n}{n}$个元素是阶为$2n$的模式中行的大小。这个序列$Q=q_1\,q_2\,q_3\cdots$是对于奇数$n>1$映射$1\mapsto 13$和$n\mapsto(n-2)n\,n(n+2)$的变换的惟一不动点，表示(36)的两步。

对于$0<x\leqslant 1$，令$f(x)=\limsup_{n\to\infty}s(\lceil xM_n\rceil)/n$。这个函数几乎在各个地方明显地消失。但当$x$有$(q_1+\cdots+q_j)/2^n$的形式时，它等于1，这是由于答案72所致。另一方面，如果我们定义$g(x)=\lim_{n\to\infty}s(\lceil xM_n\rceil)/\sqrt{n}$，则这个函数$g(x)$看来是可测量的，而且$\int_0^1 g(x)dx=\sqrt{\pi}$，尽管当$f(x)>0$时，$g(x)$是无穷的。(我们吁请给出这些猜测的严格证明或否定)。

94 77. 通过考虑蚯蚓步，由(39)得出这个提示。所以，我们可进行如下：

X1. [初始化。] 对于$0\leqslant j\leqslant n$，置$a_j\leftarrow 0$；并置$x\leftarrow 1$。(在以下步骤中，我们将有$x=1+2(a_1+\cdots+a_n)$。)

X2. [纠正尾部。] 当$x\leqslant n$时，置$a_x\leftarrow 1$和$x\leftarrow x+2$。

X3. [访问。] 访问二进位串$a_1\cdots a_n$。

X4. [是容易情况吗？] 如果$a_n=0$，则置$a_n\leftarrow 1$，$x\leftarrow x+2$，并返回X3。

X5. [寻找和推进a_j。] 置$a_n\leftarrow 0$和$j\leftarrow n-1$。当$a_j=1$时，置$a_j\leftarrow 0$，$x\leftarrow x-2$，以及$j\leftarrow j-1$。如果$j=0$则停止。否则，置$a_j\leftarrow 1$，回到X2。 ▌

78. 由(39)和习题11，为真。

79. (a) 列出诸0的下标和诸1的下标。例如，习题73中的二进位串对应于排列1 4 5 7 8 10 11 12 13 20 23 25 29 30 2 3 6 9 14 15 16 17 18 19 21 22 24 26 27 28。

(b) 使用(39)的约定，P表景在它顶部第一行有左括弧的下标和自由括弧的下标，其他下标在第二行。因此，由(38)

$$P = \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|}\hline 1&2&3&6&8&9&11&12&13&14&15&16&17&18&19&21&22&24&26&27&28\\ \hline 4&5&7&10&20&23&25&29&30\\ \cline{1-9}\end{array}.$$

[有关对于多重子集合的链的推广，参见健鹏，沃，*SIAM J. Algebraic and Discrete Methods* **2**(1981)，324-332。]

80. 这个奇怪的事实是习题79连同作者关于表景的论文中定理6一起给出的结果。参见*Pacific J. Math.* **34**(1970)，709-727。

81. 假设σ和σ'分别属于阶为n和n'的圣诞树模式中长度为s和s'的链。在这些链中至多有串的ss'对的$\min(s, s')$可以处于双杂乱中。而且，由于(39)，当把它们链接时，串的这ss'对实际上恰好组成阶为$n+n'$的圣诞树模式的$\min(s, s')$个链。因此，对于所有对的链$\min(s, s')$的和是$M_{n+n'}$，因此得出结果。我们顺便证明了不明显的恒等式

$$\sum_{j,k}\min(m+1-2j, n+1-2k)\,C_{j(m-j)}C_{k(n-k)} = M_{m+n}\ .$$

注：斯珀纳定理的这一推广独立地由吉·卡托纳*Studia Sci. Math. Hungar* **1**(1966)，59-63]和丹·克莱特曼[*Math. Zeitschrift* **90**(1965)，251-259]所证明。关于这里给出的证明和进一步的结果参见格林和克莱特曼，*J. Combinatorial Theory* **A 20**(1976)，80-88。

82. (a) 在每行m中至少有一个求值。有两个求值当且仅当$s(m)>1$，而且第一个求值产生0。因此，如果f恒等于1，我们就得到极小值M_n。如果f恒等于0，我们就得到极大值$M_n+\sum_m[sm>1]=M_{n+1}$。

(b) 设在$C_{n/2}$情况下$f(\chi(m, n/2))=0$，其中$s(m)=1$。否则，令$f(\chi(m, a))=1$，其中a由这个算法定义。当n为奇数时，这个规则意味着$f(\sigma)$总为1；但当n为偶数时，$f(\sigma)=0$当且仅当σ是该行中的第一个。(为看出为什么如此，使用以下事实，即在(41)中含σ'_j的行总有$s-2$的大小。)这个函数f确实是单调的；如果$\sigma\leqslant\tau$，以及σ有一个自由的左括弧，则τ也是。例如，在$n=8$的情况下，我们有

$$f(x_1,\cdots,x_s) = x_8 \vee x_6x_7 \vee x_4x_5(x_6 \vee x_7) \vee x_2x_3(x_4(x_5 \vee x_6 \vee x_7) \vee x_5(x_6 \vee x_7))$$ 95

(c)在这些情况下，(45)是对于所有n的解。

83. 在步骤H4中至多有3个结果是可能的——事实上，当$s(m)=1$时至多有2个。[关于更严密的界，参见习题5. 3. 4-31。在该习题的记号下，精确地有δ_n+2个单调的n个布尔变量的布尔函数。]

84. 对于这个问题我们把2^n个二进位串分划成M^n个块而不是链，其中每块的串$\{\sigma_1, \cdots, \sigma_n\}$，对于$i\neq j$，满足$\|A\sigma_i^T - A\sigma_j^T\| \geqslant 1$。然后，每块至多有一个二进位串能满足$\|A\sigma^T - b\| < \dfrac{1}{2}$。

令A'表示A的前$n-1$列，令v是第n列。假设$\{\sigma_1, \cdots, \sigma_s\}$是$A'$的一块，对下标编号使得$v^TA'\sigma_1^T$是$v^TA'\sigma_j^T$之极小者。则规则(36)定义$A$的适当块，因为我们有$\|A(\sigma_i0)^T - A(\sigma_j0)^T\| = \|A(\sigma_i1)^T - A(\sigma_j1)^T\| = \|A'\sigma_i^T - A'\sigma_j^T\|$，而且

$$\left\|A(\sigma_j 1)^T - A(\sigma_1 0)^T\right\|^2 = \left\|A'\sigma_j^T + v - A'\sigma_1^T\right\|^2$$
$$= \left\|A'(\sigma_j - \sigma_1)^T\right\|^2 + \|v\|^2 + 2v^T A'(\sigma_j - \sigma_1)^T \geqslant \|v\|^2 \geqslant 1$$

[还有更多成立；参见*Advances in Math.* **5**(1970)，155-157。这一结果扩展了约·埃·李特尔伍德和阿·西·奥佛德的一个定理*Mat. Sbornik* **54**(1943)，277-285，他们考虑了m=2的情况。]

85. 如果V有$n-m$维，我们可以重新对坐标编号使得

$$\begin{matrix}(1, & 0, & \cdots, & 0, & x_{11}, & \cdots, & x_{1m})\\ (0, & 1, & \cdots, & 0, & x_{21}, & \cdots, & x_{2m})\\ \vdots & \vdots & \ddots & \vdots & \vdots & & \vdots\\ (0, & 0, & \cdots, & 1, & x_{(n-m)1}, & \cdots, & x_{(n-m)m})\end{matrix}$$

成为一个基底，而且没有一个行向量$v_j=(x_{j1},\cdots,x_{jm})$完全为0。令$v_{n-m+1}=(-1,0,\cdots,0)$，⋯，$v_n=(0,0,\cdots,-1)$，则在$V$中0－1向量的个数是$Ax$=0的0－1解的个数，其中$A$是有列为$v_1,\cdots,v_n$的$m\times n$矩阵。但这个量至多是$\|Ax\|<\frac{1}{2}\min(\|v_1\|,\cdots,\|v_n\|)$的解的个数，由习题84它至多是$M_n$。

反过来说，对于m=1的基底，$x_{j1}=(-1)^{j-1}$产生M_n个解。[这个结果在电子选举中得到应用；参见戈勒的博士论文(斯坦福，2004)。]

86. 首先重新对4节点子树排序，使得它们的层码是0121(加上一个常数)；然后重新排序越来越大的子树直到每件事情都规范为止。得到的层码是0 1 2 3 4 3 2 1 2 3 2 1 2 0 1，而且父指针为0 1 2 3 4 3 2 1 8 9 8 1 12 0 14。

87. (a) 条件成立当且仅当对于某个k，$c_1<\cdots<c_k\geqslant c_{k+1}\geqslant\cdots\geqslant c_n$，所以，情况总数为$\sum_k\binom{n-1}{n-k}=2^{n-1}$。

(b) 注意$c_1\cdots c_k=c_1'\cdots c_k'$当且仅当$p_1\cdots p_k=p_1'\cdots p_k'$；而且在这样的情况下，$c_{k+1}<c_{k+1}'$当且仅当$p_{k+1}<p_{k+1}'$。

88. 精确地访问A_{n+1}个森林，它们中的A_k有$p_k=\cdots=p_n=0$。因此，步骤O4执行A_n次；对于$1\leqslant k<n$，在步骤O5中p_k改变$A_{k+1}-1$次。步骤O5也改变p_n总共A_n-1次。对于每个访问，如果我们在寄存器中保存p_n的话，访问内存的平均次数仅为$2+3/(\alpha-1)+O(1/n)\approx3.534$。[参见易·酷毕卡，*Combinatorics, Probability and Computing* **5**(1996)，403－417。]

89. 对于$1<k<n$，如果步骤O5置$p_n\leftarrow p_j$恰好Q_n次，它置$p_k\leftarrow p_j$恰好$Q_k+A_{k+1}-A_k$次，因为规范式$p_1\cdots p_n$的所有前缀都是规范的。我们有$(Q_1,Q_2,\cdots)=(0,0,1,2,5,9,22,48,118,288,\cdots)$；而且人们能够证明$Q_n=\sum_{d\geqslant1}\sum_{1\leqslant c<n/d-1}a_{(n-cd)(n-cd-d)}$，其中$a_{nk}$
96 是满足$p_n=k$的规范父序列$p_1\cdots p_n$的个数，但这些数a_{nk}仍保持为神秘的。

90. (a)这个性质等价于2.3.4.4－(7)；顶点0是中心点。

(b) 令$m=\lfloor n/2\rfloor$。在步骤O1结束时，置$p_{m+1}\leftarrow0$，如果n为奇数，也置$p_{2m+1}\leftarrow0$。在步骤O4结束时，置$i\leftarrow j$且当$p_i\neq0$时，置$i\leftarrow p_i$。(那么i是包含j和k的树的根)。在步

骤O5的开始，如果$k=i+m$和$i<j$，置$j\leftarrow i$和$d\leftarrow m$。

(c) 如果n是偶数，则没有$n+1$个顶点的双中心的树。否则，在$m=\lfloor n/2\rfloor$个节点上寻找正则森林的所有对$(p'_1\cdots p'_m, p''_1\cdots p''_m)$，且$p'_1\cdots p'_m \geqslant p''_1\cdots p''_m$。对于$1\leqslant j\leqslant m$令$p_1=0$，$p_{j+1}=p'_j+1$，则$p_{m+j+1}=(p''_j+m+1)[p''_j>0]$。(算法O的两个体现将生成所有这样的序列。自由树的这个算法是由弗·拉斯基和李纲给出的。请见*SODA* **10**(1999)，S939-S940。)

91. 使用以下的递归过程$W(n)$：如果$n\leqslant 2$，则返回惟一的n节点有向树。否则，选择正整数j和d使得以概率$dA_d A_{n-jd}/((n-1)A_n)$得到一个给定的对(j, d)。计算随机有向树$T'\leftarrow W(n-jd)$和$T''\leftarrow W(d)$。返回通过链接T''的j个克隆到T'的根得到的树T。[*Combinatorial Algorithms* (Academic Press，1975)，第25章。]

92. 不总是。[理·李·孔明斯，在*IEEE Trans.* CT-**13**(1966)，82-90中证明$S(G)$的图总是包含一个环。参见泊·阿·霍尔兹曼和弗·哈拉利，*SIAM J. Applied Math.* **22**(1972)，187-193。但他们的构造对于有效计算不适合，因为它们要求关于中间结果大小的奇偶性的预先知识。]

93. 是。步骤S7返工步骤S3，步骤S9返工步骤S8的删除。

94. 例如，通过一个辅助表$b_1\cdots b_n$，我们可以使用深度优先查找：

i) 置$b_1\cdots b_n\leftarrow 0\cdots 0$，然后$v\leftarrow 1$，$w\leftarrow 1$，$b_1\leftarrow 1$以及$k\leftarrow n-1$。

ii) 置$e\leftarrow n_{v-1}$。当$t_e\neq 0$时，做下列的子步骤：

a) 置$u\leftarrow t_e$。如果$b_u\neq 0$，转到子步骤(c)。

b) 置$b_u\leftarrow w$，$w\leftarrow u$，$a_k\leftarrow e$，$k\leftarrow k-1$。如果$k=0$，则终止。

c) 置$e\leftarrow n_e$。

iii)如果$w\neq 1$，则置$v\leftarrow w$，$w\leftarrow b_w$，并返回(ii)。否则报告出错：给定的图不连通。只要子步骤(b)把k减到1，我们实际上就可以终止，因为算法S决不看a_1的初始值。但我们也可以测试连通性。

95. 下列步骤从u开始实施一个宽度优先查找，看如果不使用边e是否v可达。使用弧指针的辅助数组$b_1\cdots b_n$，在步骤S1结束时它应初始化为$0\cdots 0$；我们再把它复原为$0\cdots 0$。

i) 置$w\leftarrow u$和$b_w\leftarrow v$。

ii) 置$f\leftarrow n_{u-1}$。当$t_f\neq 0$时做下列子步骤：

a) 置$v'\leftarrow t_f$。如果$b_{v'}\neq 0$，转到子步骤(d)。

b) 如果$v'\neq v$，则置$b_{v'}\leftarrow v$，$b_w\leftarrow v'$，$w\leftarrow v'$，并转到子步骤(d)。

c) 如果$f\neq e\oplus 1$，转到步骤(v)。

d) 置$f\leftarrow n_f$。

iii) 置$u\leftarrow b_u$。如果$u\neq v$，则返回步骤(ii)。

iv) 置$u\leftarrow t_e$。当$u\neq v$时，置$w\leftarrow b_u$，$b_u\leftarrow 0$，$u\leftarrow w$。转到S9。(e是一个桥)

v) 置$u\leftarrow t_e$。当$u\neq v$时，置$w\leftarrow b_u$，$b_u\leftarrow 0$，$u\leftarrow w$。然后再置$u\leftarrow t_e$，继续步骤S8(e不是一个桥)。

在开始这个计算以前可以使用两个快速的启发探索：如果$d_u=1$，则e显然是一个桥；而如果$l_e\neq 0$，则e显然不是一个桥(因为在u和v之间另有一条边)。这样的特殊 97

情况通过宽度优先查找可容易地探测，作者的实验指出，两种启发式探索都是值得的。例如，对于l_e的检测典型地节省了总运行时间的3%左右。

96. (a) 令e_k为$k-1\to k$的有向边。答案94中的步骤对于$n>k\geqslant 1$置$a_k\leftarrow e_{n+1-k}$。对于$1\leqslant k<n-1$在层k中我们收缩e_{n-k}。在访问(惟一的)生成树$e_{n-1}\cdots e_2\, e_n$之后，我们不压缩e_{n-k}，而且很快就发现，对于$n-1>k\geqslant 1$，它是一个桥。因此，运行时间是关于n线性的。作者的实现结果是，对于$n\geqslant 3$，它恰好为$102n-226$次内存访问。

然而，这个结果紧密地依赖于在初始生成树中诸边的顺序。如果当n为偶数时，在位置$a_2\cdots\ a_{n-1}$中S1已经产生了“手风琴顺序”，例如

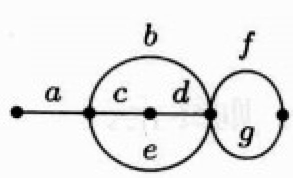
$$e_{n/2+1}\, e_{n/2}\, e_{n/2+2}\, e_{n/2-1}\cdots e_{n-1}\, e_2$$

则运行时间是$\Omega(n^2)$，因为桥测试的$\Omega(n)$中每个已花费$\Omega(n)$个步骤。

(b) 对于$n>k\geqslant 1$，a_k初始时为e_{n-k}，其中e_1是有向边$n\to 1$。当$n\geqslant 4$时，被访问的生成树分别是$e_{n-2}\cdots e_1\, e_n$，$e_{n-2}\cdots e_1\, e_{n-1}$，$e_{n-2}\cdots e_2\, e_{n-1}\, e_n$，$e_{n-2}\cdots e_3\, e_{n-1}\, e_n\, e_1$，…，$e_{n-1}\, e_n\, e_1\cdots e_{n-3}$。对于$0<k\leqslant n-4$沿着树$e_{n-2}\cdots e_{k+2}\, e_{n-1}\, e_n\, e_1\cdots e_k$计算下移到层$n-k-3$，然后再上移；桥的检测全都有效。因此，总的运行时间是二次的(在作者的版本中，对于$n\geqslant 5$，它恰好是$35.5n^2+7.5n-145$次内存访问)。

顺便指出，在斯坦福图库(SGB)的记号下，P_n是(板)*board* $(n, 0, 0, 0, 1, 0, 0)$，C_n是*board* $(n, 0, 0, 0, 1, 1, 0)$；SGB的顶点称作0到$n-1$。

97. 当$\{s, t\}$是$\{1, 2\}$，$\{1, 3\}$，$\{2, 3\}$，$\{2, 4\}$或$\{3, 4\}$时，是的。但$\{1, 4\}$不是。

98. $A'=$ 

。这是平面图A的“对偶平面图”。(A' 附近的树是A的生成树的补。反之亦然)。

99. 通过对树的大小的归纳，所述方法有效。它有效的原因实际上和它对于7.2.1.1节中的n元组有效的原因相同——但要求另外的附加条件，即我们必须成功地指定一个非省事节点的每个孩子。

叶节点总是被动的，而且它们既非省事也非不省事。所以，我们将假定在前根顺序下，分支节点编号为1到m。对于所有分支节点令$f_p=p$。但当p是一个被动不省事节点时，在它右边最接近的不省事节点是主动的；在后一情况下，f_p应指向它左边最近的活动不省事节点(为了这个定义，我们想像在左边和右边，存在人造的节点0和$m+1$，两者都不省事，都是活动的)。

F1. [初始化。] 对于$0\leqslant p\leqslant m$，置$f_p\leftarrow p$；并置$t_0\leftarrow 1$，$v_0\leftarrow 0$，置每个z_p使得$r_{z_p}=d_p$。

F2. [选择节点p。] 置$q\leftarrow m$，则当$t_q=v_q$时，置$q\leftarrow q-1$。置$p\leftarrow f_q$和$f_q\leftarrow q$；如果$p=0$，则算法终止。

F3. [改变d_p。] 置$s\leftarrow d_p$，$s'\leftarrow r_s$，$k\leftarrow v_p$和$d_p\leftarrow s'$(现在$k=v_s\neq v_{s'}$)。

F4. [更新值。] 置$q\leftarrow s$，$v_q\leftarrow k\oplus 1$。当$d_q\neq 0$时，置$q\leftarrow d_q$和$v_q\leftarrow k\oplus 1$。(现在q是一片叶，如果$k=0$，它进入该配置。如果$k=1$就不管它。)类似地，置$q\leftarrow s'$和$v_q\leftarrow k$。当$d_q\neq 0$时，置$q\leftarrow d_q$和$v_q\leftarrow k\oplus 1$(现在q是一片叶，如果$k=0$它离开配置。如果$k=1$，则它进入配置)。

F5. [访问。] 访问由所有叶的值表示的配置。 98

F6. [使p被动？] (现在p右边的所有不省事节点都成为活动的)。如果$d_p \neq z_p$，就返回到步骤F2。否则，置$z_p \leftarrow s$，$q \leftarrow p-1$；当$t_q = v_q$时，置$q \leftarrow q-1$。(现在q是p左边的第一个不省事节点，我们将使p成为被动的。)置$f_p \leftarrow f_q$，$f_q \leftarrow q$，并返回F2。 ▌

尽管步骤F4可能使不省事节点改变成省事节点，及相反，但焦点指针不必更新，因为它们仍被正确地设置。

100. 叫做GRAYSPSPAN的完整程序出现在作者的网页上。通过使用下面习题110的结果，可以证明它的渐近效率。

101. 由于通过算法与产生的顺序十分复杂，因此作者不提供。——译者注

102. 如果是这样，通常生成树可以以*强转动门顺序*列出，在每一步进入和离开的边是相邻的。

哈罗德·奈·加宝和尤金·温·迈尔斯，*SICOMP* **7**(1978)，280-287以及山·卡布尔和哈·拉密斯，*Algorithmica* **27**(2000)，120-130已经开发出生成带有给定根的所有有向生成树的有趣算法。

103. (a) 推翻以词典顺序增加$(x_0, x_1, \cdots, x_n)$，但不改变$x_0+\cdots+x_n$。如果我们能在V_i和V_j两处推翻，则两个顺序都给出相同的结果。

(b) 增加一粒沙，改变16个稳定状态如下：

给定	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
+0001	0001	0010	0011	0001	0101	0110	0111	0101	1001	1010	1011	1001	1101	1110	1111	1101
+0010	0010	0011	0001	0010	0110	0111	0101	0110	1010	1011	1001	1010	1110	1111	1101	1110
+0100	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111	0100	0101	0110	0111
+1000	1000	1001	1010	1011	1100	1101	1110	1111	0100	0101	0110	0111	1000	1001	1010	1011

递归状态是有$x_1+x_2>0$和$x_3+x_4>0$的9种情况。注意0001的重复相加导致无穷循环$0000\rightarrow0001\rightarrow0010\rightarrow0011\rightarrow0001\rightarrow0010\rightarrow\cdots$，但状态0001，0010和0011不是递归的。

(c) 如果$x=\sigma(x+t)$，那么对于所有$k\geqslant0$，也有$x=\sigma(x+kt)$。t的所有分量都是正的，因此$x=\sigma(x+\max(d_1,\cdots,d_n)t)$是递归的。反之，假设$x=\sigma(d+y)$，其中所有$y_i\geqslant0$。于是，$d+y+t$倒塌成$x+t$，它也倒塌成$\sigma(d)+y+t=d+y$。因此，$\sigma(x+t)=\sigma(d+y)=x$。

(d) 有$N=\det(a_{ij})$个类，因为初等的行操作(习题4.6.1-19)使矩阵三角化同时保持同余性。

(e) 有非负整数$m_1,\cdots,m_n$，$m_1',\cdots,m_n'$，使得

$$x+m_1a_1+\cdots+m_na_n=x'+m_1'a_1+\cdots+m_n'a_n=y$$

对于充分大的k，向量$y+kt$在$m_1+\cdots+m_n$步内倒塌成$x+kt$，而且在$m_1'+\cdots+m_n'$步内倒塌成$x'+kt$。因此，$x=\sigma(x+kt)=\sigma(x'+kt)=x'$。

(f) 在(d)中的三角化证明对于任意向量y，$x\equiv x+Ny$。而且倒塌保持同余性，因此每类有一个递归状态。

(g) 由于在平衡有向图中$a=a_1+\cdots+a_n$，我们有$x\equiv x+a$。如果x是递归的，事实上我们看到，当$x+a$减少成x时，每个顶点恰好倒塌一次，因为向量$\{a_1,\cdots,a_n\}$是线

性无关的。

反之，如果$\sigma(x+a)=x$，我们必须证明x是递归的。令$z_m=\sigma(ma)$；则必定有使$z_{m+k}=z_m$的某个正的k和m。当z_m+ka减少成z_m时，每个顶点倒塌k次。因此，有向量$y_j=(y_{j1},\cdots,y_{jn})$且$y_{jj}\geqslant d_j$，使得$(m+k)a$倒塌成$y_j$。由此得出，$x+n(m+k)a$倒塌成$x+y_1+\cdots+y_n$，而且$\sigma(x+y_1+\cdots+y_n)=\sigma(x+n(m+k)a)=x$。

99 (h) 循环地处理下标，对于$j=i_1,\cdots,i_k$带有向边$V_j\to V_0$的生成树有$n-k$个其他有向边：对于$i_l<j\leqslant i_l+q_l$的有向边$V_j\to V_{j-1}$和对于$i_l+q_l<j<i_{l+1}$的有向边$V_j\to V_{j+1}$。类似地，递归状态对于$j=i_1,\cdots,i_k$有$x_j=2$；对于$i_l<j<i_{l+1}$有$x_j=1$，除非当$j=i_l+q_l$和$q_l>0$时，$x_j=0$。

(i) 在这个情况下，状态$x=(x_1,\cdots,x_n)$是递归的当且仅当$(n-x_1,\cdots,n-x_n)$解决在这个提示中的停车问题。因为$t=(1,\cdots,1)$，而且不能停车的一个序列留下一个"孔"，它使$x+t$不倒塌成为x。

注：由迪帕克·塔尔[*Phys. Review Letters* **64**(1990)，1613-1616]引入的沙堆模型已经导致在物理参考文献中的许多论文。塔尔注意到，如果随机地引进M个粒沙，当$M\to\infty$时，每个递归状态是同概率的。本习题是在罗·科利和多·罗森的著作*European J. Combinatorics* **21**(2000)，447-459的启发下给出的。

沙堆理论证明，每个有向图D产生一个阿贝尔群，其元素对应于有根V_0的D的有向生成树。特别是，当D是一个通常的图，且每当u和v相邻，就有有向边$u\to v$和$v\to u$时，同样为真。因此，举例来说，我们可以"加"两棵生成树；而且可以认为某棵生成树为"零"。罗·科利和伊·利·博格尼，*Advances in Applied Math.* **30**(2003)，44-52已经找到，在D为一个通常图的情况下，在生成树和递归状态之间优雅的对应。但对于一般的有向图D，还不知道它们间的简单对应。例如，假设$n=2$和$(e_{10},e_{12},e_{20},e_{21})=(p,q,r,s)$，则有$pr+ps+qr$棵有向树，而且递归状态对应于推广的二维圆环体，如习题7-00中那样。而且当$p+q\geqslant s$和$r+s\geqslant q$时，甚至在"平衡的"情况下，在生成树和递归状态之间也没有明显的容易的映射出现。

104. (a) 如果$\det(\alpha I-C)=0$，则存在一个向量$x=(x_1,\cdots,x_n)^T$，使得$Cx=\alpha x$，而且，对于某个m，$\max(x_1,\cdots,x_m)=x_m=1$。于是$\alpha=\alpha x_m=c_{mn}-\sum_{j\neq m}e_{mj}x_j\geqslant c_{mm}-\sum_{j\neq m}e_{mj}=0$。(顺便指出，特征值为非负的实对称矩阵称为*半正定的*。我们的证明建立了著名的事实，即对于$1\leqslant m\leqslant n$的带有$c_{mm}\geqslant|\sum_{j\neq m}c_{mj}|$的实对称矩阵都有这个性质。)因此，$\alpha_0\geqslant 0$。而且，$\alpha_0=0$，因为$C(1,\cdots,1)^T=(0,\cdots,0)^T$。

(b) $\det(xI-C(G))=x(x-\alpha_1)\cdots(x-\alpha_{n-1})$。由矩阵树定理，$x$的系数是$(-1)^{n-1}n$乘以生成树的个数。

(c) 由习题1.2.3－36，$\det(\alpha I-C(K_n))=\det((\alpha-n)I+J)=(\alpha-n)^{n-1}\alpha$，这里$J$是元素全为1的矩阵。因此，这些外表是$0,n,\cdots,n$。

105. (a) 如果$e_{ij}=a+be'_{ij}$，我们有$C(G)=naI-aJ+bC(G')$。如果C是各行之和为零的任何矩阵，则通过将2到n的各列加到列1，并提取因式$(x+ny)/x$，接着从列2到列n减y/x乘以列1，然后从列1减去列2到列n，可以证明恒等式

$$\det(xI+yJ-zC)=\frac{x+ny}{x}z^n\det((x/z)I-C)$$

因此，通过置$x=\alpha-na$，$y=a$，$z=b$，$a=1$，以及$b=-1$，我们发现G有$0,n-\alpha_{n-1},\cdots,n-\alpha_1$这些外表。(特别是，当$G'$是空图$\overline{K_n}$时，这一结果同习题104(c)一致。)

(b) 把$\left\{\alpha_0',\cdots,\alpha_{n'-1}',\alpha_0'',\cdots,\alpha_{n''-1}''\right\}$排序。(由于多样性，这是一个容易的情况。)

(c) 这里$\overline{G}=\overline{G'}+\overline{G''}$，所以由(a)和(b)，$G'$的外表是$\{0,n'+n'',n''+\alpha_1',\cdots,n''+\alpha_{n'-1}',n'+\alpha_1'',\cdots,n'+\alpha_{n''-1}''\}$。(特别是，当$G'=\overline{K_m}$和$G''=\overline{K_n}$时，$G$是$K_{m,n}$，因此，$K_{m,n}$的外表是$\{0,(n-1)\cdot m,(m-1)\cdot n,m+n\}$。)

(d) $C(G)=I_{n'}\otimes C(G'')+C(G')\otimes I_{n''}$，其中$I_n$表示$n\times n$单位矩阵，$\otimes$表示矩阵的克罗内克乘积。$C(G)$的外表为$\{\alpha_j'+\alpha_k''|0\leqslant j<n',0\leqslant k<n''\}$。如果$A$和$B$是其特征 100
值分别为$\{\lambda_1,\cdots,\lambda_m\}$和$\{\mu_1,\cdots,\mu_n\}$的任意矩阵，则$A\otimes I_n+I_m\otimes B$的特征值是$mn$个$\lambda_j+\mu_k$的和。证明：选择$S$和$T$使得$S^-AS$和$T^-BT$是三角矩阵。然后，使用矩阵恒等式$(A\otimes B)(C\otimes D)=AC\otimes BD$来证明$(S\otimes T)^-(A\otimes I_n+I_m\otimes B)(S\otimes T)=(S^-AS)\otimes I_n+I_m\otimes(T^-BT)$。(特别是，重复使用这个公式证明，$n$立方体的外表是$\left\{\binom{n}{0}\cdot 0,\binom{n}{1}\cdot 2,\cdots,\binom{n}{n}\cdot 2n\right\}$。而且由习题104(b)，可得出等式(57)。)

(e) 当G是度数为d'的一个正则图时，它的外表是$\alpha_j=d'-\lambda_{j+1}$，其中$\lambda_1\geqslant\cdots\geqslant\lambda_n$是邻接矩阵$A=(e_{ij})$的特征值。$G'$的邻接矩阵是$A'=B^TB-d'I_{n'}$，其中$B=(b_{ij})$是有元素$b_{ij}=$[边$i$接触顶点$j$]的$n\times n'$关联矩阵，其中$n=n'd'/2$为边数。$G$的邻接矩阵是$A=BB^T-2I_n$。我们有

$$x^n\det(xI_{n'}-B^TB)=x^{n'}\det(xI_n-BB^T);$$

从以下事实可得出这个恒等式，即对于$k=1,2,\cdots$，通过牛顿恒等式(习题1.2.9-10)，借助于迹(A^k)可表达$\det(xI-A)$的系数。所以，G的外表和G'的外表相同，加上$n-n'$个外表等于$2d'$。[这一结果归功于易·博·瓦克霍夫斯基，*Sibirskiĭ Mat. Zhurnal* **6**(1965)，44-49。也参见霍·沙赫斯，*Wissenschaftliche Zeitschrift der Technischen Hochschule Ilmenau* **13**(1967)，405-412。]

(f) $A=A'\otimes A''$，所以外表为$\{d''\alpha_j'+d'\alpha_k''-\alpha_j'\alpha_k''|0\leqslant j<n',0\leqslant k<n''\}$。

(g) $A(G)=I_{n'}\otimes A''+A'\otimes I_{n''}+A'\otimes A''=(I_{n'}+A')\otimes(I_{n''}+A'')-I_n$产生外表$\{(d''+1)\alpha_j'+(d'+1)\alpha_k''-\alpha_j'\alpha_k''|0\leqslant j<n',0\leqslant k<n''\}$。

106. (a) 如果α是通路P_n的一个外表，则对于$1\leqslant k\leqslant n$，且$x_0=x_1$和$x_n=x_{n+1}$，方程组$\alpha x_k=2x_k-x_{k-1}-x_{k+1}$有非零解$(x_0,x_1,\cdots,x_{n+1})$。如果我们置$x_k=\cos(2k-1)\theta$，我们发现$x_0=x_1$且$2x_k-x_{k-1}-x_{k+1}=2x_k-(2\cos 2\theta)x_k$，因此，如果我们选择$\theta$使得$x_n=x_{n+1}$并且$x$不全为零，则$2-2\cos 2\theta=4\sin^2\theta$将是一个外表。因此，$P_n$的外表结果为$\sigma_{0n},\cdots,\sigma_{(n-1)n}$。

由习题104(b)，必定有$\alpha_1\cdots\alpha_{n-1}=n$，因为$c(P_n)=1$。因此

$$c(P_m\times P_n)=\prod_{j=1}^{m-1}\prod_{k=1}^{n-1}(\sigma_{jm}+\sigma_{kn})\cdot$$

(b, c) 类似地，如果α是环C_n的一个外表，则对于所述方程组以及$x_n=x_0$，有一个非零解。对于这个情况，我们尝试$x_k=\cos 2k\theta$并寻求对于$0\leqslant j<\lceil n/2\rceil$，$\theta=j\pi/n$时的解。而且，$x_k=\sin\theta$进一步给出对于$\lceil n/2\rceil\leqslant j<n$的线性无关的解。因此，$C_n$的外表是$\sigma_{0n}, \sigma_{2n}, \cdots, \sigma_{(2n-2)n}$，我们有

$$c(P_m\times C_n)=n\prod_{j=1}^{m-1}\prod_{k=1}^{n-1}(\sigma_{jm}+\sigma_{(2k)n}),\quad c(C_m\times C_n)=mn\prod_{j=1}^{m-1}\prod_{k=1}^{n-1}(\sigma_{(2j)m}+\sigma_{(2k)n})$$

令$f_n(x)=(x+\sigma_{1n})\cdots(x+\sigma_{(n-1)n})$以及$g_n(x)=(x+\sigma_{2n})\cdots(x+\sigma_{(2n-2)n})$。这些多项式有整系数。其实，$f_n(x)=U_{n-1}(x/2+1)$和$g_n(x)=2(T_n(x/2+1)-1)/x$，其中$T_n(x)$和$U_n(x)$是由$T_n(\cos\theta)=\cos n\theta$和$U_n(\cos\theta)=(\sin(n+1)\theta)/\sin\theta$定义的切比雪夫多项式。$c(P_m\times P_n)$的计算可归结为一个$m\times m$行列式的计算，因为它是$f_m(x)$和$f_n(-x)$的结式。参见习题4.6.1-12。类似地，$\dfrac{1}{n}c(P_m\times C_m)$和$\dfrac{1}{mn}c(C_m\times C_n)$分别是$f_m(x)$和$g_n(-x)$及$g_m(x)$和$g_n(-x)$的结式。

令$\alpha_n(x)=\Pi_{d\backslash n}f_d(x)^{\mu(n/d)}$，这样$\alpha_1(x)=1$，$\alpha_2(x)=x+2$，$\alpha_3(x)=(x+3)\times(x+1)$；$\alpha_4=x^2+4x+2$，$\alpha_5(x)=(x^2+5x+5)(x^2+3x+1)$，$\alpha_6(x)=x^2+4x+1$。通过考虑所谓的域多项式，
101 人们能够证明，当n是偶数时，$\alpha_n(x)$是整数不可约多项式，否则，它是相同次的两个不可约因式的乘积。类似地，如果$\beta_n(x)=\Pi_{d\backslash n}g_d(x)^{\mu(n/d)}$，结果是当$n\geqslant 3$时$\beta_n(x)$是不可约多项式的平方。这些事实说明了在结果中相当小质因子存在的原因。例如，对于$m\leqslant n\leqslant 10$在$c(P_m\times P_n)$中最大的质因子是1009。它仅出现在$\alpha_6(x)$和$\alpha_9(-x)$的结式中，它是$662913=3^2\cdot 73\cdot 1009$。

107. 对于$n=(1, \cdots, 5)$，有$(1, 1, 2, 6, 21)$个异构图，但我们只需考虑有$\leqslant\frac{1}{2}\binom{n}{2}$条边的情况，因为习题105(a)所致。当$n=4$时幸存的情况是自由树：星形是$K_1+K_3$的补，相对于0, 1, 1, 4；通过习题106，P_4有0, $2-\sqrt{2}$, 2, $2+\sqrt{2}$的外表。当$n=5$时，有三棵自由树。星形有0, 1, 1, 1, 5的外表；P_5的外表是0, $2-\phi$, $3-\phi$, $1+\phi$, $2+\phi$； 的外表为0, r_1, 1, r_2, r_3，其中$(r_1, r_2, r_3)\approx(0.52, 2.31, 4.17)$是$x^3-7x^2+13x-5=0$的根。

最后，有五个带有一个环的情况：是$K_1\,\overline{+}\,(K_2+\overline{K_2})$。所以，它的外表是0, 1, 1, 3, 5；$C_5$有外表0, $3-\phi$, $3-\phi$, $2+\phi$, $2+\phi$；有外表0, r_1, r_2, 3, r_3；它的补有外表0, $5-r_3$, 2, $5-r_2$, $5-r_1$；而且的外表结果是0, $(5-\sqrt{13})/2$, $3-\phi$, $2+\phi$, $(5+\sqrt{13})/2$。

108. 给定顶点$\{V_1, \cdots, V_n\}$上的有向图D，令e_{ij}是从V_i到V_j的有向边的个数。像以前一样定义$C(D)$和它的外表，由于$C(D)$不必是对称的，则其外表不再保证为实的。但是，如果α是一个外表，则它的复共轭$\overline{\alpha}$也是。如果通过它们的实部来对外表排序，我们再次发现$\alpha_0=0$。如果解释$c(D)$为对所有可能的n个根V_j取平均的有向生成树的平均个数，则公式$c(D)=\alpha_1\cdots\alpha_{n-1}/n$保持正确。对于$1\leqslant i<j\leqslant n$，其有向边为$V_i\rightarrow V_j$的传递淘汰赛$T_n$的外表明显地是0, 1, $\cdots$, $n-1$；而且它的子图的外表同样是明显的。

答案105的部分(a)~(d)的推导可不变地进行。例如，考虑$K_1 \mp T_3$，它有外表0, 2, 3, 4；这个有向图D有(2, 4, 6, 12)个有向生成树且有四个可能根，而且$c(D)$确实等于2 · 3 · 4/4。并且注意有向图是它自己的补，而且它和T_3有相同的外表。

有向图也允许另一族有趣的运算：如果D' 和D'' 是不相交顶点集合V' 和V''上的有向图，每当$v' \in V'$ 和$v'' \in V''$时，考虑加a条有向边$v' \to v''$ 和b条有向边$v'' \to v'$。通过像在答案105(a)中那样操作行列式，我们可以证明，得到的有向图有外表 $\{0, an'' + bn', an'' + \alpha_1', \cdots, an'' + \alpha_{n'-1}', bn' + \alpha_1'', \cdots, bn' + \alpha_{n''-1}''\}$ 。在特殊情况a=1和b=0下，我们通过$D' \to D''$ 可以方便地表示新的有向图。例如，$T_n = K_1 \to T_{n-1}$。在$n_1+n_2+\cdots+n_m$个顶点上有向图$K_{n1} \to K_{n2} \to \cdots \to K_{nm}$ 有外表$\{0,\ n_m \cdot s_m,\ \cdots,\ n_2 \cdot s_2,\ (n_1-1) \cdot s_1\}$，其中$s_k = n_k + \cdots + n_m$。

从V_1到V_n的有向通路Q_n 的外表明显地是0, 1, ⋯, 1。有向环O_n有外表$\{0, 1-\omega, \cdots, 1-\omega^{n-1}\}$，其中$\omega = e^{2\pi i/n}$。

有向边的有向图也有一个漂亮的结果：D^* 的外表可通过对于$1 \leqslant k \leqslant n$，简单地加数$\sigma_k$的$\tau_{k-1}$个副本由$D$而得到，其中$\tau_k$是$V_k$的入度而$\sigma_k$是它的出度。(如果$\tau_k$=0，我们删去等于$\sigma_k$的一个外表。)这个证明类似于答案2.3.4.2-21中的推导，但比它还简单。

历史注记：习题104(b)和105(*a*)的结果归功于阿 · 柯 · 克尔曼斯，*Avtomatika i Telemekhanika* **26**(1965)，2194-2204；27,2(1966年2月)，56-65；*Automation and Remote Control* **26**(1965)，2118-2129中的英文翻译；**27**(1966)，233-241。米罗斯 102
拉夫 · 菲耶德尔，[*Czech. Math. J.* **23**(1973)，298-305]介绍了习题105(d)并证明了关于外表α_1的有趣结果。他称为G的“代数连通性”。格尔曼 · 克利维拉斯，*J. Combinatorial Theory* **B24**(1978)，202-212枚举了栅格，圆柱体，圆环体的生成树的个数，以及在诸如$O_m \times O_n$ 这样的有向圆柱体上有向生成树的个数。博延 · 莫哈尔在*Graph Theory，Combinatorics and Applications* (Wiley：1991)，871-898；*Discrete Math.* **109**(1992)，171-183上发表了图的外表的杰出评述。关于图的特征值的重要族和性质的透彻的讨论包括综合参考书目，参见由德 · 默 · 斯维科赫维克，迈 · 杜布和赫 · 沙赫斯所著*Spectra of Graphs*，第三版(1995)。

109. 也许也有一个与沙堆有关的原因，参见习题103。

110. 通过归纳法；假设在u和v之间有$k \geqslant 1$条平行的边，则$c(G) = kc(G_1) + c(G_2)$，其中G_1是带有标识u和v的G，而G_2是删去k条边的G。令$d_u = k + a$和$d_v = k + b$。

情况1：G_2是连通的。则$ab > 0$，所以，我们可以写$a = x + 1$和$b = y + 1$。我们有$c(G_1) > \alpha\sqrt{x+y+1}$和$c(G_2) > \alpha\sqrt{xy}$，其中$\alpha$是其他$n-2$个顶点的乘积，而且容易验证

$$k\sqrt{x+y+1} + \sqrt{xy} \geqslant \sqrt{(x+k)(y+k)}$$

情况2：没有这样的u和v使G_2连通。则G的每一个多边是一个桥。换言之，G除了平行边外是一棵自由树。在这个情况下，如果有度为1的顶点，则结果是平凡的。否则，假设u是一个端点，而且有$d_u = k$条边u——v。如果$d_v > k+1$，我们有$c(G) = kc(G_1) > \alpha k\sqrt{x}$，其中$d_v = k + 1 + x$，容易验证当$x > 0$时，$k\sqrt{x} > \sqrt{(k-1)(k+x)}$。

如果$d_v=k$，我们有 $c(G)=k>\sqrt{(k-1)^2}$。最后，如果$d_v=k+1$，令$u_0=u$，$v_1=v$，并考虑惟一的通路v_1——v_2——⋯——v_r，其中$r>1$，而且v_r有大于2的度；对于$1\leqslant j<r$，仅有一条边连接v_j到v_{j+1}。再次通过归纳法获得证明。

[关于生成树的个数的其他下界，已经由阿·瓦·科斯多彻卡推导出来，*Random Structures and Algorithms* **6**(1995)，269－274。]

111. 2 1 5 4 11 7 9 8 6 10 15 12 14 13 3。

112. 或者p出现在一个偶层上，而且是q的一个祖先，或者q出现在一个奇层上，而且是p的一个祖先。

113. 前后根顺序(F^R)=后前根顺序$(F)^R$和后前根顺序(F^R)=前后根顺序$(F)^R$。

114. 考虑到森林可能为空，最优雅的方法是把事物朝上，使得CHILD(Λ)指向最左树的根，如果有的话。然后，通过置Q←Λ，L←－1，初始化第一个访问，并转向步骤Q6。

115. 假设在偶层有n_e个节点和在奇层有n_o个节点，而且偶层节点的n_e'个是无叶的。则步骤(Q1, ⋯, Q7)分别执行$(n_e+n_o, n_o, n_e', n_e, n_e', n_o+1, n_e)$次，由于答案114，因此它包括对于Q6的一次执行。

116. (a) 这一结果从算法Q得出。

(b) 事实上，非寻常节点严格地在幸运和不幸运之间交替，并且从幸运的节点

102 开始和结束。证明：考虑由删去F的最左叶得到的森林F'，并且对n使用归纳法。

117. 这样的森林精确地是：其左儿子/右兄弟表示是一棵退化的二叉树的那些

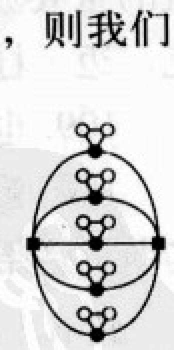
103 (习题31)。所以答案为2^{n-1}。

118. (a) 对于$k>1$，t^{k-2}。幸运仅出现在接近极端的叶。

(b) 一个有趣的递归得到解$(F_k+1-(k+1)\bmod 3)/2$。

119. 以值$v(x)=\sum\{2^k \mid k$是从根到x的通路上一个有向边的标号$\}$标记每个节点x。于是在前后根顺序下节点的值恰好是格雷二进制码Γ_n，因为习题113证明了它们满足递归式7.2.1.1-5。

(如果我们应用相同的值标号到通常的二项式树T_n和以前根顺序遍历，则我们就只得到整数0, 1, ⋯, 2^n-1。)



120. 假的。在一个哈密顿环中，在图中的“空洞”顶点中仅有四个可出现在靠近两个“方”顶点。因此一对空洞是由幸运节点出来的。[参见赫·费莱斯尼尔和哈·范·克隆克，*Monatshefte für Mathematik* **76**(1972)，112-117。]

121. 其次，在T^2中从u到v有一条哈密顿通路当且仅当类似的条件成立；但我们保留在$T^{(1)}$中的u和/或v，如果它们有度1的话，而且我们要求(i)中的通路在从u到v的通路里边(不包括u和v本身)。条件(ii)也通过把“度4的顶点”改变成“危险的顶点”而加强，其中$T^{()}$的一个顶点叫做危险的，如果它有度4，或者有度2而且等于u或v。最小的不可能情况是$T=P_4$且u和v选择为非端点。[*Časopis pro Pěstování Matematiky* **89**(1964)，323-338。]

结果，T^2包含一个哈密顿环当且仅当T是毛虫，即其推导是一条通路的自由树。[参见弗朗克·哈拉里和艾·约·斯温克，*Mathematika* **18**(1971)，138-140]。

122. (a) 通过一棵二叉树，我们可表示一个表达式，而且运算符在内部节点而数字在外部节点。如果像在算法B中那样实现二叉树，则由给定的文法所强加的实质性限制是，如果$r_j=k>0$，则在节点j的运算符是+或－当且仅当在节点k的运算符是×或/。因此，具有n个叶的一棵树的可能性总数是$2^n S_{n-1}$，其中S_n是一个施罗德数，即当n=9时，为10 646 016。(参见习题66，但交换左与右。)我们可以更快地把它们全部生成出来，恰好遇到1640个解。仅一个表达式，即1+2/((3－4)/(5+6)－(7－8)/9)不用乘法来做这件事。它们中的20个，例如(((1－2)/((3/4)×5－6))×7+8)×9，要求5对括弧；仅有15个不要求任何括弧。

(b) 现在有 $1+\sum_{k=1}^{8}\binom{8}{k}2^{k+1}S_k$ =23 463 169种情况以及3365个解。杜德尼发现了长度为12的最短者[*The Weekly Dispatch* (18 June 1899)]，即123－45－67+89。但在当时，他并不确信它是最好的。最长的解长度为27。如上所述，它们共有20个。

(c) 情况数急剧地增加到 $2+\sum_{k=1}^{8}\binom{8}{k}4^{k+1}S_k=8\,157\,017\,474$，而且现在有96 504个解。作为惟一的最长者有长度40：((((. 1/(. 2+. 3))/. 4)/. 5)/(. 6－. 7))/(. 8－. 9)。有5个有趣的例子，诸如. 1+(2+3+4+5)×6+7+8+. 9，有7个加号。其次，有10个如(1－. 2－. 3－4－. 5－6)×(7－8－9)这样的，它们共有7个－号。

> 事实上在此事中有很少的原理，
> 也没有确定的方法证明我们已获得
> 最好可能的解。
> ——亨利 · 恩 · 杜德尼(1899)

注：玛利 · 莱斯克最初于1864年出版的*Illustriertes Spielbuch für Mädchen*，包含了这个问题最早的有名表述。在第11版(1889)中，100=1+2+3+4+5+6+7+8×9是对于553节中难题16的解。也参见习题7. 2. 1. 1-111的参考文献。 104

理查德 · 贝尔曼在*AMM* **69**(1962)，640-643中说明如何处理部分(a)的特殊情况，其中运算符限制为+或者×，不带括弧。他的动态规划技术也可用于这个更一般的问题来减少考虑的情况。想法是确定从数字{1, …, n}的每个子区间可得到的有理数，并且有一个给定的运算符在树根。通过删去不能导致整数解的子区间{1, …, 8}和{2, …, 9}的情况，我们也可节省大量的计算。这样一来，要考虑的本质上不同的树的个数减少到(a) 2 747 275种情况；(b) 6 834 708；(c) 741 167 401。

浮点算术在这个应用中是不可靠的，但4.5.1节的精确有理数算术子程序漂亮地做这个工作，决不需要以绝对值大于10^9的整数来做。

123. (a) 2284；但2284=(1+2×3)×(4+5×67)－89。(b) 6964；但6964=(1/.2)×34+5+6789；(c) 14786；但14786=－1+2×(.3+4+5)×(6+789)。[如果我们也允许在表达式左边使用负号，如同杜德尼所做的那样，实际上得到对于问题120(a)，(b)

和(c)的1361，2758和85054个另外的解，包括在情况(a)中19个更长的表达式，例如$-(1-2)\times((3+4)\times(5-(6-7)\times 8)+9)$。通过这样的推广，在现有问题中最小不可达的个数变为(a) 3802，(b) 8312和(c) 17722。]可表示整数(正，负或零)的全部个数是(a) 27 666，(b) 136 607，(c) 200 765。

124. 霍特顿－斯特拉勒数源于对河流流动的研究。罗·艾·霍特顿，*Bull. Geol. Soc. Amer* **56**(1945)，275-370；阿·尼·斯特拉勒，*Bull. Geol. Soc. Amer* **63**(1952)，1117-1142。由维恩诺特，艾罗勒斯，简尼和阿尔奎斯所写的经典文章，
105 *Computer Graphics* **23**，3(1989年7月)，31-40剖析和说明了许多画树的思想。

7.2.1.7 节

1. 也许在六线形中的21表现为“嘎吱嘎吱响”(䷔)；然而，古代的评论员更多地把这个六线形同法律的实施关联起来，而不是和电子的交互作用相关联。

2. a) 对于在密码中的第一个核苷酸，令(T，C，A，G)分别用(⚌,⚍,⚎,⚏)表示，类似地用(⚌,⚍,⚎,⚏)表示第二个核苷酸。用(⚌,⚍,⚎,⚏)表示第三个；并且重迭这三个表示。因此，六线形数34是䷡ = ⚌ + ⚌ + ⚍；它表示密码子TTC，它映射到氨基酸F。在这个对应下，六线形34到54(含)分别映射到值(F, G, L, Q, W, D, S, −, P, Y, K, A, I, T, N, H, M, R, V, E, C)。而且，映射到“−”的三个六线形是数1, 9和41，即䷀，䷈和䷨，在易经中意味着“创造”，“平淡”和“消除剩余”——全都十分适合于完成一个蛋白质的组成的思想。

b) 考虑$\binom{64}{6,6,6,4,4,4,4,4,4,3,3,2,2,2,2,2,2,2,2,2,1,1}\approx 2.3\times 10^{69}$种方法来排列$4\times 4\times 4$基因编码数组的元素。它们中刚好有

$$240288040217578979000399368196455132845166871875018555392000000\approx 2.4\times 10^{63}$$

个至少包含21个不同的连续元素的一个游程。[使用容斥原理能够证明，有r个不同元素和$n_r=0$的任何多重集合$\{(n_1+1)x_1,\cdots,(n_r+1)x_r\}$恰好有

$$(n+1)\binom{n}{n_1,\cdots,n_r}r!-\sum_{k=1}(n+1-k)k!(r-k)!a_k\sum_{\substack{0\le d_1,\cdots d_r\le 1\\ d_1+\cdots+d_r=k}}\binom{n-k}{n_1-d_1,\cdots,n_r-d_r}$$

个这样的排列，其中$n=n_1+\cdots+n_r$，a_k是有k个元素的不可分解排列的个数(习题7.2.1. 2-100)。] 因此，在每百万个排列中大约仅有一个有所述性质。

但是，有$4!^3\binom{6}{2,2,2}$=1 244 160种方式来表示像部分(a)中那样的密码子。它们中的大多数对应于氨基酸的不同排列(除在第三个位置中交换T和C的表示外)。

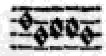
根据经验，事实上，64个六线形的所有排列大约有31%证实有适当的密码子映射。因此，部分(a)的构造没有给出原因使人们相信，易经的作者们以任何方式预先处理了基因编码。

3. 由于$F_{31}-10^6=F_{28}+F_{22}+F_{20}+F_{18}+F_{16}+F_{14}+F_9$，因此第100万个是

⌣⌣⌣⌣⌣⌣⌣—⌣⌣⌣—————⌣⌣⌣⌣—⌣⌣.

到另一条路是容易的：$F_{31}-(F_5+F_8+F_{10}+F_{16}+F_{18}+F_{27}+F_{30})=314\ 159$。

4. 在行4 כמדיר 的两个出现之一应该是 כמידר；这个小故障可能只不过是一个印刷错误。类似地，在行8的一个 ירדמכ 应该是 ימדרכ，但在行3和行4，最右字母 כד 的六种情况出现两次；而最右 כד 的情况却不见了。唐诺罗本人必须对于这个差错负责。

5. 最后一个应该是 ，而不是 。

6. 对于$1\leqslant n\leqslant 13$和$15\leqslant n\leqslant 38$，在梅森表中的第$n$个值$m_n$仅同$n!$一致。梅森知道$14!=87178291200\neq m_{14}=8778291200$，因为他在该书的个人保存副本中插入了丢失的“1”(现在该书归Bibliothèque Nationale所有，其摹本出版于1975年)。但是在他的表中的其他错误不仅是印刷上的，因为它们传播到随后的元素中。而m_{50}的情况例外：$m_{39}=39!+10^{26}-10^{10}$；$m_{40}=40m_{39}$；$m_{41}=41m_{40}-4\cdot 10^{25}-14\cdot 10^{11}$；对于 106
n=42, 43, 44, 46, 47, 48, 49, 55, 60和62，$m_n=nm_{n-1}$；$m_{50}=50m_{49}+10^{66}$；$m_{51}=51\cdot 50m_{49}$。当他计算$m_{45}=9\cdot 45\cdot m_{44}-10^{40}+10^{29}$时，他明显地想采用一条捷径，因为乘以5或乘以9是容易的；但他两次乘以9。大多数他的错误指出一个不可靠的乘法技术，这可能是同算盘有关：$m_{52}=52m_{51}+5\cdot 10^{56}-2\cdot 10^{47}+10^{34}$；$m_{53}=53m_{52}-4\cdot 10^{29}$；$m_{54}=54m_{53}+10^{16}$；$m_{57}=57m_{56}+10^{33}+10^{24}$；$m_{58}=58m_{57}+10^{67}-10^{35}+10^{32}+11\cdot 10^{26}$；$m_{59}=59m_{58}+10^{66}+10^{49}-10^{28}$；$m_{61}=61m_{60}-5\cdot 10^{81}$；$m_{63}=63\cdot m_{62}+10^{82}-10^{74}$；$m_{64}=64m_{63}+3\cdot 10^{81}+10^{67}+2\cdot 10^{38}-2\cdot 10^{33}-10^{23}$。

剩下的情况，$m_{56}\approx 10.912m_{55}$是令人迷惑的，它是$\equiv 56m_{55}$(modulo 10^{17})。但它的其他数字似乎既不满足韵律也没有理由。能容易地说明它们吗？

注：阿坦纳西乌斯·克彻尔在他的*Ars Magna Sciendi* (1669)一书的157页对于$1\leqslant n\leqslant 50$造出的$n!$表必定是从梅森的书中摘录这些结果，因为他重复了梅森的所有错误。然而，克彻尔确实列出$10m_{14}$，$m_{45}/10$和$10m_{49}$的值代替m_{14}，m_{45}和m_{49}。也许他试图使这个序列增长得更平稳些。不清楚是谁首先正确地计算出39!的值。习题1. 2. 5-4讲述了1000!的故事。

7. 基本的排列是12345，13254，14523，15432，12453，14235，15324，13542，12534，15243，13425，14352。但我们发现，所有60个偶排列既是活的也是死的，因为(9)和(8)相差一个偶排列。(而且，如果稍微修改n=5的情况，对于n=6有一半的活排列将转成为奇的)。

8. 例如，我们可以以

$$a_n a_3\cdots a_{n-1}a_2 a_1,\ a_1 a_{n-1}\cdots\ a_n a_3 a_2,\ \cdots,\ a_{n-1}a_2\cdots\ a_{n-2}a_1 a_n$$

代替(9)，这样就触发了末尾，并且循环地移动(8)的排列中的其他元素。这个修改有效，因为所有排列都有正确的奇偶性，而且活的和死的在每个可能位置中都有a_1。(如等式7.2.1.2-(32)中交替的组，我们实际上有一个对偶的Sims表，但我们的元素叫做$(n, n-1, \cdots, 1)$，而不是$(0, 1, \cdots, n-1)$。

埃·毕佐特发表了生成有适当符号的排列的一个更简单的方法[*Mémoires Acad. Royale des Sciences* (巴黎，1764)：292]，$\{1, \cdots, n-1\}$的每个排列$a_1\cdots\ a_{n-1}$产生n个其他的，$a_1\cdots\ a_{n-1}a_n-a_1\cdots\ a_{n-2}a_n a_{n-1}+\cdots$。

9. (٠,١,٢,٣,٤,٥,٦,٧,٨,٩)；也许我们应该说：(٩,٨,٧,٦,٥,٤,٣,٢,١,٠)。注：方程组的下标数使用一个不同的系统；例如，“ر”代表200。而且，还应注意，(11)实际上是阿尔萨玛瓦尔的著作的誊本翻译成现代阿拉伯文；阿赫玛德和拉塞德把它们的工作建立在14世纪的副本基础上，它使用类似的但较老的的数字形式：(٥,١,٢,٣,۴,8,٦,٧,٨,٩)。阿尔萨玛瓦尔本人可能也使用了一个甚至更早版本的数字。

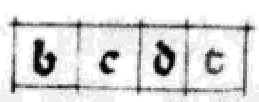
10. 如果56种情况是同样可能的，则如在票券搜集器问题(习题3.3.2-8)中一样，答案将是$56H_{56}\approx 258.2$。但(6,30,20)的情况分别以概率(1/216, 1/72, 1/36)出现；正确的答案证实为

$$\int_0^\infty (1-(1-e^{-t/216})^6(1-e^{-t/72})^{30}(1-e^{-t/36})^{20})\,dt \approx 546.6$$

这大约是上界$216H_{216}$的42%。[请参见菲·弗拉约勒特，丹·加蒂和罗·提莫尼尔，*Discrete Applied Math.* **39**(1992)，207-229。]

11. 一次从(b, c, d, B, C, D)中取三个的$\binom{6}{3}=20$种组合造表。如果我们认为

107 b<c<d<B<C<D，则它们以词典顺序出现。字母t(t)意味着“从小写移到大写”[参见安·博恩纳，*Selected Works of Ramon Llull* (Princeton，1985)，596-597]。有两种打印错误，在第6行开始处“d”应是“b”；在18行结尾处的“c”应是“d”。如果勒拉尔已经把它表示为

；

则行1应该同其他行更为一致。但在该行，当然不需要有大小写的转移。

12. 以5(mod 7)乘泊恩索特的循环。

13. 当有n个不同字母时，最好恰有n行

$$\frac{a.\ aa.\ aaa}{b.\ ab.\ aab.\ aaab.\ bb.\ abb.\ aabb.\ aaabb}$$

那么，如在(18)中，赋权$(a, b)=(1,4)$给出数1到11。((16)的第一行也应省略。)类似地，对于(a,a,a,b,b,c)，我们将含蓄地给c以权12，并且加上另外的行：

c. ac. aac. aaac. bc. abc. aabc. aaabc. bbc. abbc. aabbc. aaabbc.

[雅·伯努利在*Ars. Conjectandi*，第2部分第6章中几乎正确地给出它。]

14. ABC ABD ABE ACD ACE ACB ADE ADB ADC AEB AEC AED BCD BCE BCA BDE BDA BDC BEA BEC BED BAC BAD BAE CDE CDA CDB CEA CEB CED CAB CAD CAE CBD CBE CBA DEA DEB DEC DAB DAC DAE DBC DBE DBA DCE DCA DCB EAB EAC EAD EBC EBD EBA ECD ECA ECB EDA EDB EDC。它是一个广义的词典顺序(参见算法7.2.1.3R)，并且通过未使用的字母周期地进行。

[在*Sha ‘ari Tzedeq*为标题的犹太神秘哲学的著作中，使用类似的顺序形成五个字母的所有120个排列，这归功于13世纪的作者西西里的默西纳，纳丹·本·沙阿迪亚·哈拉尔。参见*Le Porte della Giustizia*(米兰：Adelphi，2001)。]

15. 在j之后，重复地放置$\{j,\cdots,m\}$的$(n-1)$组合，所以，答案为$\binom{(m+1-j)+(n-1)-1}{n-1}=\binom{m+n-j-1}{n-1}$。[吉安·波利尔，也称作布特奥尼斯，在他早期的书*Logistica*(里昂：1560)的305-309页指出这一点。对于$1\leqslant n\leqslant 4$，他对于n个骰子的所有投掷造表，然后对于j求和，演绎出，对于n=5，有56+35+20+10+4+1=126种不同的投掷，以及对于n=6，此数为462。]

16. N1. [初始化。] 置$r\leftarrow n$，$t\leftarrow 0$，以及$a_0\leftarrow 0$。

N2. [推进。] 当$r\geqslant q$时，置$t\leftarrow t+1$，$a_t\leftarrow q$，以及$r\leftarrow r-q$。如果$r>0$，则置$t\leftarrow t+1$，以及$a_t\leftarrow r$。

N3. [访问。] 访问合成$a_1\cdots a_t$。

N4. [求j。] 置$j\leftarrow t$，$t-1$，$\cdots$，直到$a_j\neq 1$为止。如果j=0，则终止此算法。

N5. [减少a_j。] 置$a_j\leftarrow a_j-1$，$r\leftarrow t-j+1$，$t\leftarrow j$，返回N2。 ▌

例如，对于n=7和q=3合成是331, 322, 3211, 313, 3121, 3112, 31111, 232, 2311, 223, 2221, 2212, 22111, 2131, 2122, 21211, 2113, 21121, 21112, 211111, 133, 1321, 1312, 13111, 1231, 1222, 12211, 1213, 12121, 12112, 121111, 1132, 11311, 1123, 11221, 11212, 112111, 11131, 11122, 111211, 11113, 111121, 111112, 1111111。

纳拉燕纳的79节和80节基本上给出这个过程，但是有颠倒的串(133, 223, 1123, …)，因为他喜欢递减的协词典顺序。[沙恩加第瓦，在*Saṅgītaratnākara* §5. 316-375中，以前已建立起对于所有合成(韵律)集合的一个精心作出的理论，可以由基本部分{1, 2, 4, 6}形成。]

17. 访问的个数V_n是$F^{(q)}_{n+q-1}=\Theta(\alpha_q^n)$；参见习题5.4.2-7。步骤N4测试$a_j$=1的次数$X_n$满足$X_n=X_{n-1}+\cdots+X_{n-q}+1$，我们求得$X_n=V_0+\cdots+V_n=(qV_n+(q-1)V_{n-1}+\cdots+V_{n-q+1}-1)/(q-1)=\Theta(V_n)$。步骤N2置$a_t\leftarrow q$的次数$Y_n$满足相同的递归式，我们求得$Y_n=X_{n-q}$。而且步骤N2求$r$=0的次数结果是$V_{n-q}$。 108

18. 在罗马数字中，它是MDCLXVI，其中M>D>C>L>X>V>I。

19. 行329和1022。(普提亚奴斯包括139个这样的短诗到他1022的表中。)

20. “tria”居于“lumina”之前，一个扬抑抑格有5!×2!×(11, 12, 12, 16)种方式分别居于(第1个，第2个，第3个，第4个)音步；“lumina”居于“tria”之前有5!×2!×(16, 12, 12, 11)种方式。所以总数为24 480。[莱布尼茨在他的*Dissertatio de Arte Combinatoria*接近末尾处考虑了这个问题，而且得出答案45 870，但他的论证中有错误。]

21. (a) 生成函数$1/((1-zu-yu^2)(1-zv-yv^2)(1-zw-yw^2))$显然等于$\sum_{p,q,r,s,t\geqslant 0} f(p,q,r;s,t)\,u^p v^q w^r z^s y^t$。

(b) 如果“tibi”是⌣⌣而‘Virgo’是——，则此数是3!3!乘以$\sum_{k=0}^{3}(f(2k+1, 6-2k, 2; 3.3)+f(2k, 6-2k, 2; 2, 3))$，即36((7+7)+(9+5)+(10+5)+(14+7))=2304。否则“tibi”是⌣—，“Virgo”是—⌣，此数为2!3!乘以$\sum_{k=0}^{3}(f(2k, 5-2k, 2; 3, 2)+f$

(2k, 6 − 2k, 1; 3, 2))，即12((7+6)+(5+4)+(4+4)+(0+6))=432。

(c) 第5个音步开始于“caelo”，“dotes”或“Virgo”的第二个音节。因此，另外一个数为3!3! $\sum_{k=0}^{2} f(2k, 5-2k, 2; 3, 2)=36(7+5+4)=576$，而且总数为2304+432+576 =3312。

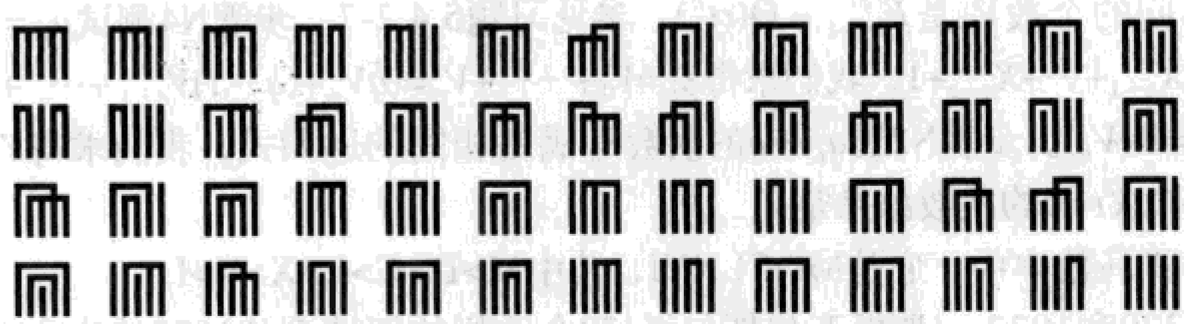
22. 令$\alpha\in$\{quot, sunt, tot\}，$\beta\in$\{cælo dotes Virgo\}，σ=sidera和τ=tibi。普勒斯特的分析实质上等价于伯努利的分析，但他忘记包括36种情况$\alpha\alpha\alpha\tau\beta\beta\sigma\beta$。(按他的风味，人们可以说，那些情况在诗歌中是老一套。普提亚奴斯没发现它们有什么用。)普列斯特的书(1675年版本)也省略了以$\tau\beta$结尾的所有排列。

沃利斯把可能性分为23种类型，$T_1\cup T_2\cup\cdots\cup T_{23}$。他断言，他的类型6和7，每个生成324个短诗。但实际上，$|T_6|=|T_7|=252$，因为他的变量i应是7，而不是9。他也把许多解统计了2次：$|T_3\cap T_5|=72$，$|T_2\cap T_7|=|T_5\cap T_7|=|T_3\cap T_6|=|T_6\cap T_{10}|=36$，$|T_{11}\cap T_{12}|=|T_{12}\cap T_{13}|=|T_{14}\cap T_{15}|=12$。他疏忽了36种可能性$\alpha\beta\beta\alpha\sigma\alpha\tau\beta$ (其中的19个为普提亚奴斯所用)。他还疏忽了习题20(c)的所有排列。普提亚奴斯使用了576个当中的250个。出版于1693年的沃利斯的书的拉丁文版本改正了这一部分的许多印刷错误，但没有改正数学上的错误。

惠特沃思和哈特利省略了带有“tibi”= ◡— 的所有情况(参见习题18)。可能是因为人们对于经典的六进制的知识开始衰退。

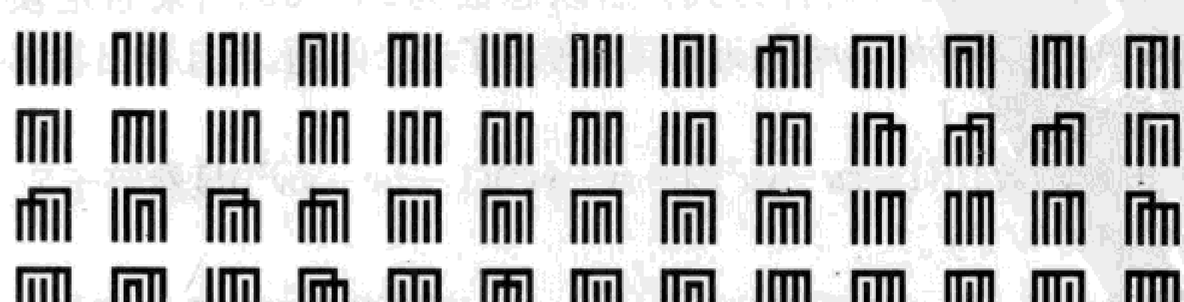
[谈及错误，普提亚奴斯实际上仅发表了1020个不同的排列，而非1022个，因为在他的表中的592行和593行等同于601行和602行。但是他应毫无麻烦地找出另外两种情况——例如，通过将行252, 345, 511, 548, 659, 663, 678, 693或797中的“tot sunt”改为“sunt tot”]。

23. 从左到右地阅读每个框图，使得12|345↔ ，我们得到



109

24. 他的规则是，对于$k=0, 1, 2, \cdots, n-1$，以及$n-1$个事物中一次取k个的每个组合$0<j_1<\cdots<j_k<n$，访问$\{1, \cdots, n-1\}\backslash\{j_1, \cdots, j_k\}$的所有分划和块$\{j_1, \cdots, j_k, n\}$。对于$n=5$，他的顺序是：



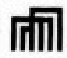
但是严格地说，这个习题的答案是“否”——因为本田利明的规则在确定组合的顺序之前是不完备的。他生成在协词典顺序的组合(按$j_i\cdots j_1$的词典顺序)。$j_1\cdots j_i$的词典顺序也将同对于n=4给出的表一致，但他将把  放在 之前。参考文献：

林鹤一，*Tôhoku Math. J.* **33**(1931)，332-337。

25. 否：(28)疏忽1 4|2 3 5。(它的第二个模式的顶部−底部的反射)。

26. 令a_n是{1, ⋯, n}的不可分解分划的个数，a'_n是不可分解且完备的分划的个数。这些序列以$\langle a_1, a_2, \cdots\rangle = \langle 1, 1, 2, 6, 22, 92, 426, \cdots\rangle$，$\langle a'_1, a'_2, \cdots\rangle = \langle 0, 1, 1, 3, 9, 33, 135, \cdots\rangle$开始。对于$n\geqslant 2$，这个习题的答案是$a'_n - 1$。它证实，$a_n$也是在不可交换的变量中次数为$n$的对称多项式的个数。[参见马·卡·沃尔弗(M. C. Wolf), *Duke Math. J.* **2**(1936)，626-637，他还把不可分解的分划造表成k个部分。]

如果$A(z)=\sum_n a_n z^n$，$B(z)=\sum_n \varpi_n z^n$是贝尔数的非指数生成函数，我们有$A(z)B(z)=B(z)-1$。因此，$A(z)=1-1/B(z)$。习题7.2.1.5-35的结果意味着$\sum_n a'_n z^n = zA(z)/(1+z-A(z)) = z(B(z)-1)/(1+zB(z))$。不幸的是，$B(z)$没有特别好的封闭形式。注意，对于$n>1$的不可分解的集合分划对应于振动的表循环，而且没有三个连续的λ等于0(参见习题7.2.1.5-27)。

27. 问题是二义性的，因为源氏-香框图没有很好地定义。我们要求，一个块区的所有垂直线有相同的高度。那么，1 4 5|2 3 6没有单个交叉的框图因为不允许。

没有交叉的分划个数是C_n (参见习题7.2.1.6-26)。对于一个交叉，交叉的两个块的元素必定交叉出现于限制的增长序列内。如$x^i\, yx^j\, y^k$或$x^i\, y^{j+1}\, xy^k$或者$x^i\, y^j\, xy^k\, x^l$，其中，$i, j, k, l>0$。

假设模式为$x^i\, yx^j\, y^k$。在这个模式的$i+l+j+k$个元素之间有$t=i+j+k+2$个“点”，而且以非交叉的分划来填入这些点的方式数为$\sum_{i_1+\cdots+i_t=n-i-j-k-1} C_{i_1}\cdots C_{i_t}$。通过等式7.2.1.6-(24)，我们可以把这个数表达为

$$[z^{n-i-j-k-1}]\,C(z)^{i+j+k+2} = C_{(n-i-j-k-1)n}$$

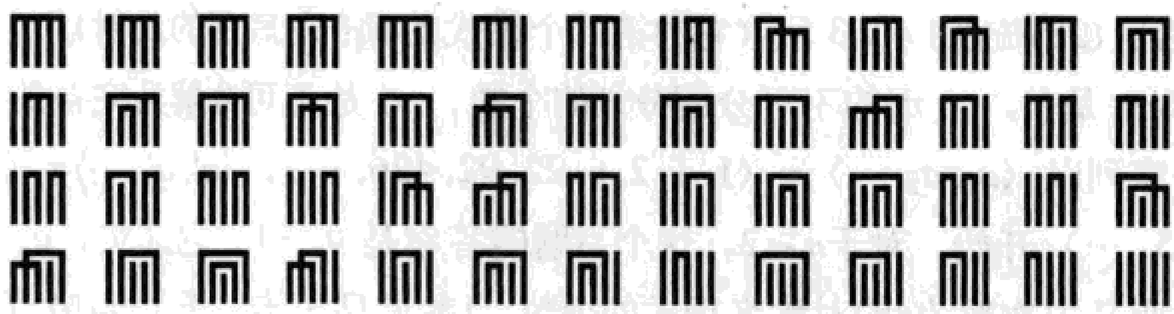
对k求和，给出$C_{(n-i-j-2)(n+1)}$；然后对于j和i求和，给出$C_{(n-4)(n+3)}$。

类似地，其他两个模式给出$C_{(n-5)(n+3)}$和$C_{(n-5)(n+4)}$。因此，单个交叉分划的总数为$C_{(n-5)(n+3)}+C_{(n-4)(n+4)}$。

28. 通过质因子的个数来对$cbbaaa$的因子排序，而后在协词典顺序下排成为：$1\prec a\prec b\prec c\prec aa\prec ba\prec ca\prec bb\prec cb\prec aaa\prec baa\prec caa\prec bba\prec cba\prec cbb\prec baaa\prec caaa\prec$
$bbaa\prec cbaa\prec cbba\prec bbaaa\prec cbaaa\prec cbbaa\prec cbbaaa$。在递减顺序下，对于每个这样 110
的因子d，令d是第一个因子，递归地附加其第一个因子$\preceq d$的$cbbaaa/d$的所有因子分解。

如果诸因子已经以词典排序(即$1<a<aa<aaa<b<ba<\cdots<cbbaa<cbbaaa$)，则沃利斯算法将等价于$(n_1, n_2, n_3)=(1, 2, 3)$的算法7.2.1.5M。他大概选择这些因子更复杂的顺序，因为它趋向于同当$a\approx b\approx c$时通常的数值顺序更为一致。例如，当$(a, b, c)=(7, 11, 13)$时，他的顺序精确地是数值的。按照他更复杂的方案来生成这些因子，沃利斯实际上生成多重集合的组合，我们已在7.2.1.3节中说明，它们等价于有界的合成。[参考文献：*A Discourse of Combinations* (1685)，126-128，有两个已改正的印刷错误。]

29. 因式分解$edcba$, $edcb \cdot a$, $edca \cdot b$, …, $e \cdot d \cdot c \cdot b \cdot a$分别对应于



30. 除非$i_1+2i_2+\cdots=n$，否则这个系数为0。在该情况下，它是$\binom{m}{k} a_0^{m-k}\binom{k}{i_1,i_2,\ldots}$，其中$k=i_1+i_2+\cdots$。(考虑$(a_0z)^m$乘以$(1+(a_1/a_0)z+(a_2/a_0)z^2+\cdots)^m$。)

31. 由该算法产生的顺序是递减的词典顺序，即(31)的颠倒，如果我们假定分划$a_1\cdots a_k$有$a_1\geqslant\cdots\geqslant a_k$的话；德·模弗的顺序是递增的协词典顺序。

32. $20\cdot1=7+13\cdot1=2\cdot7+6\cdot1=10+10\cdot1=10+7+3\cdot1=2\cdot10$。一般，博斯科维奇建议以$n\cdot1$开始，计算$a\cdot10+b\cdot7+c\cdot1$的后继者如下：如果$c\geqslant7$，则后继者是$a\cdot10+(b+1)\cdot7+(c-7)\cdot1$；否则，如果$c+7b\geqslant10$，则后继者是$(a+1)\cdot$
111 $10+(c+7b-10)\cdot1$；否则停止。

索引和词汇表

索引中的页码为英文原书页码，与书中页边标注的页码一致。当一个索引指向包含相关习题的一页时，有关进一步的信息也请参见该习题的答案。这里对答案页不作索引，除非它指向的内容未包括在该习题的陈述中。

0-1 vectors（0-1向量）,42

μ, 参见Mems

$\nu(x)$：The number of 1s(=Sideways sum)（1的个数(=旁路和)）

π(circle ratio),as “random” example（作为“随机”的例子的圆周率）,21, 41, 48, 89, 106

A

Abacus（算盘），107

Abelian group（阿贝尔群），100

Adachi，Fumie（安达文江），66

Adjacency matrix（邻接矩阵），101

Ahmad，Salah(صلاح أحمد)（阿赫玛德 · 沙拉），55，107

al-Samaw'al(=as-Samaw'al)（阿尔沙马瓦尔）ibn Yaḥyā ibn Yahūda al-Maghribī (المغربي السموءل ابن يحيى بن يهودا)，（伊本 · 雅亚，伊本 · 雅 · 胡塔，阿尔马格里比），55，107

Aldous，David John（阿尔多斯 · 戴维 · 约翰），16

Algebraic connectivity（代数连通性），103

Aliquot parts（整除部分），67

Alonso，Laurent（阿伦索 · 劳伦），92

Alternating group（交替群），107

Amino acids（氨基酸），73

Anagrams（变位子），53

Analysis of algorithms（算法分析），12，38，42，45，74，81

Ancestor，in a tree structure（树结构中的祖先），6，76，103

Antichain of subsets（子集的反链），21

Arabic mathematics（阿拉伯数学），55，61

Arabic numerals（阿拉伯数字），55，74

Arborescences，树形图，参见Oriented trees

Arc digraph of a digraph（有向图的弧边有向图），102

Archimedes of Syracuse（Ἀρχιμήδης ὁ Συρακούσιος）（锡拉库萨的阿基米德，固体），solids，37

Arima，Yoriyuki（有马赖僮），66

Aristotle of Stagira，son of Nicomachus(Ἀριστοτέλης Νικομάχου ὁ Σταγιρίτης)（斯塔吉利亚的亚里士多德，尼古马彻斯的儿子），58

Aristoxenus（Ἀριστόξενος）（阿里斯托泽奴斯），52

Arnold，David Bryan（阿诺德，戴维 · 布里安），15

Arquès，Didier（阿尔奎斯，迪迪尔），105

Aspects of a graph（图的外表），44-45

Associahedron（相关联边），37

Associative law（结合律），9，34，37，46

Atkinson，Michael David（阿特金森，迈克尔 · 戴维），92

Atomic strings of parentheses（括弧的原子串），40

Attributes of God（上帝的属性），56-58

Avalanches（雪崩），44

B

Backtrack method（回溯方法），65

Balanced digraphs（平衡的有向图），44

Balanced strings（平衡的串），40

Baldéric（巴尔德里克），56

Ballot numbers（选票数），13-14，38，110

generalized（广义），38-39
table（表），13
Baronaigien，*see* Roelants van Baronaigien（巴尔罗奈吉安，参见鲁兰特·范·巴尔罗奈吉安）
Bauhuis，Bernard(=Bauhusius，Bernardus)（鲍休斯，贝尔纳德(=贝尔纳度斯)），62-64
Beckenbach，Edwin Ford（贝克金巴赫，欧文·福的），72
Becker，Harold W.（贝克，哈罗德W.），81
Bell，Eric Temple，numbers（贝尔，埃里克·坦普尔，数），67，110
Bellhouse，David Richard（贝尔豪斯，戴维·里查德），56
Bellman，Richard Ernest（贝尔曼，里查德·厄内斯特），105
Bernoulli，Jacques(=Jakob=James)（伯努利，雅格奎斯(=雅各布=詹姆斯)），48，61，64-65，108，109
Beyer，Wendell Terry（贝叶尔，温德尔·特利），23
Bézout，Étienne（贝佐特，埃提恩尼），107
Bhāskara II，Ācārya，son of Māheśvara (भास्कराचाय，माहेश्वरपुत्र)（巴斯卡拉 II，阿卡里亚·马赫斯瓦拉的儿子），53，54
Bhattotpala(=Utpala，भट्टोत्पल),)（巴托特帕拉(=乌特帕拉),)，62
Biclutters（双杂乱），42
Bigraph:A bipartite graph（双部图：双部分的图）
Binary arithmetic（二进制算术），49
Binary search（二分搜索），22
Binary search trees（二叉查找树），39，47
Binary trees（二叉树），3-11，33-41，71-73
decorated（装饰），18，40
degenerate（退化），37，39，103
drawings of（画），3，16，17，47，71，77，81
extended（扩展），3，18，34，39，47
Gray codes for（格雷码），8-11，35
linked（链接），6-11，35
random（随机），18-19，40，47
representation of（表示），6，10，18，38，40，73
rotations in（转动），9-11，82
Binomial coefficient identities（二项式系数恒等式），39
Binomial trees（二项式树），46，104
Bonner，Anthony Edmonde（博恩纳，安托尼·埃德蒙德），108
Boolean functions（布尔函数），22，42
Boolean lattices（布尔格），83
Borrel，Jean(=Buteonis，Ioannes)（波利尔，吉安(=布提奥尼斯，伊约安尼斯)），108
Bošković, Ruđer Josip (Бошковић, Руђер Јосип =Boscovich, Ruggiero Giuseppe=Boscovich, Roger Joseph)（博斯科维克，鲁德尔·约希普(=博斯科维奇，罗杰尔·约瑟夫)），69，75
Bounded compositions（有界组成），111
Bourgogne -Artois, Jeanne de（有界人的阿特托伊斯，吉安尼·德），58
Branch nodes: Nonleaves（分支节点·非叶），28-29
Breadth-first search（宽度优先查找），97
Bridges of a graph（图的桥），26-27，43，46，98，103
Brown，Charles Philip（布朗，查尔斯·菲利浦），51
Brown，Robert，*see* Brownian excursion（布朗，罗伯特，参见布朗短游程）
Brownian excursion（布朗短程游，布朗运动），16
Bruijn，Nicolaas Govert de（布鲁因，尼古拉斯·科维特·笛），19
cycles（循环），51
Burkhardt，Johann Carl（巴克哈特，约翰·卡尔），70
Buteonis，Ioannes(=Borrel，Jean)（布提奥尼斯，伊安里斯(=波勒尔，吉安)），108

C

C_n (Catalan number)（C_n(卡塔兰数)），12-15，18，38-39，71，73，92，110
C_n(cyclic graph)（C_n(循环图)），43，98，101
C_{pq} (ballot number)（C_{pq} (选票数)），13-14，21，38-39，110
Callan，Columcille David（卡尔兰，科鲁姆西

利·戴维），78
Canonical forest（规范森林），23-24，42-43
Canonical form of algebraic expression（代数表达式的范式），46
Cantor，Moritz Benedikt（康托尔，莫利兹·本尼迪克特），70
Carlitz，Leonard（卡尔里兹，利奥纳德），88
Cartesian product of graphs（图的笛卡儿积），v，29，45
Catalan，Eugène Charles（卡塔兰，尤金·查尔斯），12
Catalan numbers（卡塔兰数），12-15，18，38-39，71，73，92，110
generalized（推广的），38-39
tables（表），12-13
Catalan triangle（卡塔兰三角），13-14，21，38-39
t-ary（*t*进制），85，90
Caterpillar graphs（毛虫图），104
Cayley，Arthur（凯莱，阿瑟），71
Centroid of an oriented tree（有向树的心点），43
Chains of submultisets（子矩阵链），95
Chains of subsets（子集链），19-23
Characteristic polynomial of a matrix（矩阵的特征多项式），44，101
Chebyshev，Pafnutii Lvovich（Чебышев，Пафнутий Львович）polynomials（切比雪夫，帕弗奴提·勒沃维奇，多项式），101
Chinese mathematics（中国数学），48-49
Chorees（扬抑格），51-52
Christ，Wilhelm von（克利斯特，威廉·冯），52
Christian mathematics（基督教数学），55-65，74
Christmas tree pattern（圣诞树模式），19-23，41-42，94
Chung，Kai Lai（钟开莱），92
Clutters（杂乱），21
Co-atoms（协原子），40
Codons（密码子），73
Coforests（协森林），10-11
Cognate forests（同类森林），34
Colex order:Lexicographic from right to left（协词典顺序：从右至左的词典顺序），7，35，50，62，74，108，110，111
Combinations（组合），1，5，8，54-62，70，74，79，86，107，110
of a multiset（多重集合的），74，111
with repetition（带有重复），55-56，61，74
Complement of a graph（图的补），45，102
Complementary elements of a lattice（格的补元素），37
Complements（补），48，51
Complete bipartite graphs（完备的双部分图），59，100
Complete graphs（完全图），43，44，57-60
Complete rhyme schemes（完全韵律方案），75
Complete *t*-ary trees（完全的*t*叉树），46
Complete ternary trees（完全的三叉树），49
Complex conjugates（复共轭），102
Compositions:Ordered partitions（组成：有序的分划），50，54，74，111
Compression（压缩），58
Concentric wheels（同心轮），59
Concordant bit strings（和谐位串），42
Configs（配置），29，43
Confucius（孔子(孔丘=孔仲尼=孔子)），48
Conjugate of a forest（森林的共轭），10，33-34，38，45
Conjunction of graphs（图的连接），v，45
Connected graphs（连通图），26，32，46
Connectivity test（连通性测试），97，102
Context-free grammars（上下文无关文法），46
Continuant polynomials（持续的多项式），70
Contraction of a graph（图的紧凑化），25-27，93
Coproduct of graphs（图的协积），v
Cori，Robert（科利，罗伯特），100
Cosum of graphs（图的协和），v，45
Coupon collector's problem（收集券问题），107
Covering in a lattice（在一个格中的覆盖），19，35-38
Crests，Japanese heraldic（克利斯特斯，日本的纹章学的），66

Crossings in a set partition（在一个集合分划中的交叉），35，75

Cube of a graph（图的立体），32

Cummins，Richard Lee（康明斯，理查德·李），97

Cvetković，Dragoš Mladen (Цветковић, Драгош Младен),（克维特科维格，德拉戈斯·姆拉登），30，103

Cycle Lemma（循环引理），40

Cyclic graph (C_n)（循环图(C_n)），43，98，101

Cyclic permutations（循环排列），38

Cyclic shifts（循环移位），40

Cylinder graphs（圆柱体图），31，45，103

D

Dactyls（扬抑抑格），51，63，74

Dancing links（跳舞链接），26

Data structures for graphs（图的数据结构），26-28

de Bruijn Nicolaas Govert（德·布鲁因，尼戈拉斯·戈维特），19

  cycles（循环），51

de Moivre，Abraham（德·莫弗，阿布拉罕），13，68-70，75

de Montmort，Pierre Rémond（德·蒙特莫特，皮利·利莫德），68，75

Decorated binary trees（装饰二叉树），18，40

Defect of a balanced string（平衡串的缺失），40

Degenerate binary trees（退化的二叉树），37，39，103

Degree of a node（节点的度），34

Degree of a vertex（顶点的度），26，45

Degree one，nodes of（度数1的节点），28，41

  vertices of（顶点），29，46，103

Dénes，József（登尼斯，约泽基夫），84

Deo，Narsingh (नरसिंह देव),（迪奥，纳尔辛格），72

Depth coordinates c_k（深度坐标c_k），6，23-24，33，35，39，42，81

Depth-first search（深度优先查找），97

Derivative of a graph（图的导数），46

Dershowitz，Nachum (נחום דרשוביץ)（德尔索维茨，纳纯），84

Descendant，in a tree structure（树结构中的后裔），6，76，103

Descents of a permutation（排列的下降），41

Determinants（行列式），54，100-102

Deutsch，Emeric（丢特斯，埃米立克），93

Dewey，Melvil，notation for binary trees (due to Galton)（德威，麦尔维尔，二叉树的记号(由高尔顿给出)），89

  notation for trees（树的记号），14

Dhar，Deepak (दीपक धर)（达尔，迪帕克），100

Diagonally dominant matrix（对角占优矩阵），100

Dice（骰子），55-56，67，68，74，108

Digital Century puzzle（数字世纪难题），46

Digraph:A directed graph（有向图），44-45

Diomedes (Διομήδης),（狄俄墨得斯），51

Direct product of graphs（图的直接积），v，45

Directed graphs（有向图），44-45

Directed torus graphs（有向圆环体图），103

Distributive laws（分配律），36

Divisors（因子），60，67

DNA（脱氧核糖核酸），73

Donnolo，Shabbetai ben Avraham (שבתי בן אברהם דונולו)（多诺罗，沙比太·本·阿华拉罕姆），52，74

Doob，Michael（杜布，迈克尔），103

Doubly linked lists（双重链接表），26-27

Drawing a binary tree（画一个二叉树），3，16，17，47，77，81

Drexel(=Drechsel=Drexelius)，Jeremias (=Hieremias)（德利塞尔（=德里彻塞尔=德里西里尤斯），吉利米亚斯（=Hieremias)），52-53

Dual of a forest（森林的对偶），10-11，34-36，79

Dual of a planar graph（平面图的对偶），98

Dudeney，Henry Ernest（杜迪尼，亨利·厄内斯特），46，104-105

Dyck，Walther Franz Anton von（迪克，沃尔塞尔·弗朗兹·安顿·冯），72

  paths，*see* Nested parentheses. words（通路，参见嵌套括弧，字），72

Dynamic programming（动态规划），105

E

e，as “random” example（作为“随机”例子），37
Easy nodes（省事节点），29，43
Ebbenhorst Tengbergen，Cornelia van（埃布本霍斯特 · 藤伯根，科尼利亚 · 范），19
Edelman，Paul Henry（埃德尔曼，鲍尔 · 亨利(=鲍罗)），84
Eigenvalues（特征值），44-45，103
Elements(earth，air，fire，water)（元素(土地，空气，火，水)），48，59
Ellipses（椭圆），17
Empty graph（空图），100
Empty set（空集），61
Empty string（空串），93
Endo-order（端点顺序），8
Endpoint of a graph（图的端点），46，103
Equivalence classes（等价类），44
Er，Meng Chiau（余明昭），79
Erdős，Pál(=Paul)（厄尔多斯 · 鲍罗），71
Errera，Alfred（厄尔利拉，阿尔弗里德），76
Errors（错误），49，52-54，64-65，67，73-74，109，111
Ettingshausen，Andreas von（埃特廷绍森，安德利亚斯 · 冯），70
Euler，Leonhard (Eйлеръ, Леонардъ = Эйлер, Леонард)（欧拉，伦纳德），69
Eulerian numbers（欧拉数），41
Eulerian trails（欧拉迹），60
Even permutations（偶排列），54，74
Exact cover problem（精确覆盖问题），65
Extended binary trees（扩展的二叉树），3，18，34，39，47
Extended ternary trees（扩展的三叉树），34
External nodes（外部节点），3，17，18，34，39，47，76，90
Eyrolles，Georges（艾罗勒斯，乔治斯），105

F

Factorial number system（阶乘数系），62
Factorials，tables of（阶乘，表），61，74
Factoring an *n*-cycle（*n*循环阶乘），38
Factorizations of an integer（整数的因式分解），67，75
Feller，Willibald(=Vilim=Willy=William)（费勒，威利巴尔德(威利姆=威利=威廉)），92
Feussner，Wilhelm（浮斯纳，威廉），24
Fibonacci，Leonardo，of Pisa(=Leonardo filio Bonacci Pisano)，numbers（斐波那契，比萨的利奥那多（=利奥那多 · 菲利奥 · 波那契 · 比萨诺)，数），50，106
  generalized（广义），50，108
Fibonacci trees（斐波那契树），46，47
Fiedler，Miroslav（菲耶德尔，米罗斯拉夫），103
Field polynomials（域多项式），102
Flajolet，Philippe Patrick Michel（菲拉约勒特，菲律比 · 帕特里克 · 迈克尔），107
Flavors（风味韵味），54
Fleischner，Herbert（费莱斯尼尔，希尔伯特），104
Flip permutations（触发排列），84
Floating point arithmetic（浮点算术），105
Focus pointers（焦点指针），29，43
Fontana Tartaglia，Niccolò（方塔纳 · 塔尔塔格里亚，尼科罗），61
Footprints（足迹），37，83
Forests（森林），2-47，71
  canonical（典范式的），23-24，42-43
  cognate（同类），34
  conjugate of（共轭），10，33-34，38，45
  dual of（对偶），10-11，35，79
  Gray codes for（格雷码），8-11，35
  oriented（有向），23-24，42
  outline of（概述），15
  random（随机），40
  representation of，*see*（表示，参见）Left-child/right-sibling（左儿子/右兄弟）
    links，Nested parentheses（链接，嵌套的括弧），
    Right-child/left-sibling links（右儿子/左兄弟链接）
  roots of（的根），3

shape of（形状），15-16
super-root of（超根），32，45
transpose of（转置），33-34，38
triply linked（三重链接），32-33，42，80
Fractal dimension（分形维数），41
France，queen of（法国的皇后），58
Free groups（自由群），72
Free parentheses（自由括弧），21，94，95
Free trees（自由树形），24，42-43，46，71，84，102，103
Freese，Ralph Stanley（弗里西，拉尔芙·斯坦利），84

G

Gabow，Harold Neil（加宝，哈罗德·奈尔），99
Galilei，Galileo（伽利略，伽里略），67
Galton，Francis（高尔顿，弗朗西斯），113
Games（游戏），55-56，65，74
Gardner，Martin（加德纳，马丁），59
Gardy，Danièle（加蒂，丹尼尔），107
Generalized Catalan numbers（广义卡塔兰数），38-39
Generating functions（生成函数），12，38，39，41，43，75，88-89，93，109，110
Genetic code（基因码），73
Genji-ko（源氏香），65-67，71，75
Genlex order（广义词典顺序），108
Gershwin，George（吉尔斯文，乔治），60
God（上帝），56-58
Golle，Philippe（戈勒，菲律比），42，96
Gradenigo，Pietro（格拉迪尼戈，皮特罗），57
Grammar，context-free（文法，上下文无关），46
Graphs（图），24-32，43-46
Grätzer，George（格拉特泽尔，乔治），83
Gray，Frank，codes（格雷，弗朗克，码），32，72
binary（二进制），46，93
for binary trees（二叉树），8-11，35
for combinations（组合），8
for forests（森林），8-11，35
for nested parentheses（嵌套的括弧），8，35，39，82
for permutations（排列括弧），60
for Schröder trees（施罗德树括弧），41
for spanning trees（生成树括弧），25
for trees（树形括弧），8-11
modular（模括弧），43
quasi-（拟括弧），31-33
reflected（反射门括弧），9
revolving-door（转动门括弧），8，25，29，30，43
Greatest lower bound（最大下界），36-37
Greedy algorithm（贪婪算法），50
Greek poetry（希腊诗歌），51，63
Greene，Curtis（格林，库尔梯斯），21，95
Grid graphs（栅格图），31，45，103
triangular（三角形），82
Gutjahr，Walter Josef（古特雅尔，沃尔特·约瑟夫），91

H

Hakimi，Seifollah Louis（哈基米，塞福拉·路易斯），73
Hamilton，William Rowan，cycle（哈密顿，威廉·罗万，循环），104
path（通路），46，82
Hamiltonian graph:A graph that contains a Hamiltonian cycle（哈密顿图：包含哈密顿环的图），32，46
Hammond，Eleanor Prescott（汉默蒙德，埃里亚诺·珀利斯戈特），56
Handshaking at a circular table（循环表中的握手），6，33
Hansel，Georges（汉塞尔，乔治），22-23，42
Har'ar，Natan ben Sa ʻadyah (נתן בן סעדיה הרר)（哈拉尔，纳丹·本·沙阿迪亚），108
Harary，Frank（哈拉里，弗朗克），71，97，104
Hariharan，Ramesh(ரமேஷ் ஹரிஹரன்)（哈里哈兰，拉密斯），99
Hartley，William Ernest（哈特利，威廉·内斯特），64-65，75

Hayashi, Tsuruichi（林鹤一）, 110
Hebrew letters（希伯莱字母）, 52, 59, 106, 108
Hedetniemi, Sarah Lee Mitchell（黑迪特尼米，沙拉·李·米彻尔）, 23
Hexagrams（六线形）, 48-49, 73, 106
Hexameter（六步韵）, 62-65, 74
Hickerson, Dean Robert（希克森，迪安·罗伯特）, 38
Hindenburg, Carl Friedrich（辛登伯格，卡尔·弗雷德里茨）, 69-70
Hindu mathematics（印度数学）, 49-51, 53-54, 61-62, 69, 74
Hodges, Joseph Lawson, Jr.（小霍德格斯，约塞夫·劳森）, 92
Holzmann Poisson, Carlos Alfonso（霍尔茨曼·泊松，卡罗斯·阿尔方索）, 97
Homer (῞Ομηρος)（荷马）, 63
Honda, Toshiaki（本田利明）, 66, 75
Horton, Robert Elmer（霍尔顿，罗伯特·埃尔默）, 47, 105
Horton-Strahler numbers（霍尔顿-斯特拉勒数）, 47

I

I Ching（易经）, 48-49, 73-74
Iambuses（伊亚姆布瑟斯）, 51-52, 63, 74
Ibn Mun 'im (ابن منعم)（伊本·姆恩伊姆）, 61
Idel, Moshe(משה אידל)（埃德尔，莫瑟）, 59
Identity matrix（单位矩阵）, 100-101
In-degree of a vertex（顶点的入度）, 44, 102
Incidence matrix（关联矩阵）, 101
Inclusion and exclusion（容斥）, 106
Indecomposable partitions（不可分解的分划）, 75
Indecomposable permutations（不可分解的排列）, 106
Indian mathematics（印度数学）, 49-51, 53-54, 61-62, 69, 74
Indian numerals（印度数字）, 53
Inorder(symmetric order)（中根顺序(对称顺序)）, 3, 9, 10, 17, 38, 39, 73, 77, 82, 83
Integer partitions（整数分划）, 67-70, 75
Internal path length（内部通路长度）, 39
Internet（互联网）, ii, iii, 73, 99
Inverse of a permutation（排列的逆）, 10
Inversion tables（反演表）, 6, 33, 76
Islamic mathematics（伊斯兰数学）, 55, 61
Isthmuses, *see* Bridges of a graph（伊斯特姆瑟斯，参见图的桥）
Izquierdo, Sebastián（伊泽奎尔多，施巴斯蒂安）, 60-61, 74

J

Janey, Nicolas（简尼，尼古拉斯）, 105
Japanese mathematics（日本数学）, 54, 65-67
Jesus of Nazareth, son of Joseph (בן יוסף בן נצרת ישוע, Ἰησοῦς ἀπὸ Ναζαρὲθ υἱὸς Ἰωσήφ)（纳扎利思的耶稣，约瑟夫的儿子）, 53
Join of graphs（图的并）, v, 45
Juxtaposition of graphs（图的连接）, v, 45
Juxtaposition of forests（森林的连接）, 34

K

*k*th power of a graph（图的*k*次乘幂）, 32
Kabbalah（希伯莱神秘哲学或神秘教义）, 52, 59, 108
Kak, Subhash Chandra (सुभाष चन्द्र काक)（卡克，苏巴斯·哈戈德拉）, 51
Kaplansky, Irving（卡普兰斯基，伊尔文）, 71
Kapoor, Sanjiv (संजीव कपूर)（卡布尔，山吉夫）, 99
Katona, Gyula(Optimális Halmaz)（卡托纳，吉马拉(奥普梯马里斯·哈尔玛兹)）, 95
Kedāra Bhaṭṭa (केदार भट्ट)（克达拉·巴塔）, 50, 69
Keil, Heinrich（基尔，海恩里兹）, 52
Kelmans, Alexander Kolmanovich (Кельманс, Александр Кольманович)（克尔曼斯，阿历山大，柯尔曼诺维奇）, 102
Kemp, Rainer（克姆普，赖纳）, 89, 91
Kimono（和服(日本服饰)）, 66
King Wen of Chou（姬昌=周文王）, 48-49, 74

Kircher，Athanasius（克彻尔，阿塔纳西尤斯），52，53，59，74，107
Kirschenhofer，Peter（克斯承霍夫尔，彼得），91
Klee，Victor La Rue，Jr.（小克里，维克多·拉·鲁维），iii
Kleitman，Daniel J(Isaiah Solomon)（克列特曼，丹尼尔·J(伊赛亚·索罗门)），21，42，95
Kleppis，Gregor(=Kleppisius，Gregorius)（克里皮斯，格里高尔），74
Klügel，Georg Simon（格鲁吉尔，乔格·西蒙），70
Knobloch，Eberhard Heinrich（克诺布罗赫，埃贝尔哈德·海恩里兹），67，68
Knott，Gary Don（克诺特，加里·唐），73
Knuth，Donald Ervin（高德纳，克努特·唐纳德·欧文，本书著者），i，ii，iv，19，95，98，99
Korobkov，Vitaly Konstantinovich (Коробков, Виталий Константинович)（柯罗波科夫，维塔利·康斯坦第诺维奇），23
Korsh，James，F（科尔斯，詹姆斯），6，35，80，81，93
Kostochka，Alexandr Vasilievich (Косточка, Александр Васильевич)（科斯多彻卡·阿历山大·瓦西里耶维奇），103
Kreher，Donald Lawson（克里赫，唐纳德·劳森），73
Kreweras Germain（克利维拉斯，格尔曼），81，82，103
lattice（格），35-38，83
Kronecker，Leopold，product of matrices（矩阵的克罗尼克，利奥皮尔德乘积），100
Kronk，Hudson Van Etten（克隆克，哈德森·范·埃特廷），104
Kruskal，Joseph Bernard，Jr.，function（小克鲁斯卡尔，约瑟夫·贝尔纳德函数），39
Kruyswijk，Dirk（克鲁斯维依克，迪尔克），19
Kubicka，Ewa（酷毕卡，易瓦），96
Kusuba，Takanori（楠叶隆德），62

L

Lakser，Harry（拉克塞尔，哈里），83
Lambert，Johann Heinrich（兰伯特，约翰·海恩里兹），69
Latin poetry（拉丁诗歌），62-65，74
Lattices of trees（树的格），35-38
Le Borgne，Yvan Franccoise André（里波尔根，伊凡·弗朗索斯·安德列），100
Leaf nodes（叶节点），28，35，37，81
Least upper bound（最小上界），35-37
Left path length（左通路长度），89
Left-sibling/right-child links（左兄弟/右儿子链接），7，34，38，78，81-82
Left-child /right-sibling links（左儿子/右兄弟链接），3，7，9，32，77，78，81-82，103
Lehmer，Derrick Henry（莱默，德里克·亨利），72
Leibniz，Gottfried Wilhelm，Freiherr von（莱布尼茨，戈特菲里德·威廉·弗利赫尔·冯），49，63-64，67-68，74，109
Leske，Marie（莱斯克，玛利），104
Level coordinates c_k（层次坐标c_k），4，6，14，23-24，33，35，36，39，42，81
Lévy，Paul（列维，保罗），16
Lexicographic order（词典顺序），4-5，21，23，33，39，42，49，53，55，56，61，67，68，70，71，108，111
Li，Gang(=Kenny)（李钢），97
Line graph of a graph（图的边图），45
Linked binary trees，generation of（链接二叉树，生成），6-11，35
random（随机），18-19，40
Littlewood，John Edensor（李特尔伍德，约翰·埃登索），96
Llull，Ramon(=Lullus，Raimundus)（勒拉尔，拉蒙(=鲁尔拉斯，莱蒙杜斯)），56-59，74
London，John（伦敦，约翰），58
Loopless algorithm（无循环算法），32
Loops in a graph（图中的循环），24，27
Louchard，Guy（劳查德，盖伊），16
Lucas，Joan Marie（卢卡斯，佐安·玛利），9
Lucky nodes（幸运节点），45-46
Luczak(=Łuczak)，Malwina Joanna（卢兹扎克，玛尔维纳·佐安纳），93

Ludus Clericalis（鲁达斯，克里利卡里斯），55，74
Lydgate，John（里德吉特，约翰），56
Lynn，Richard John（林，理查德 · 约翰），49

M

M_n (middle binomial coefficient)（M_n(中间的二项式系数)），19，41-42
MacMahon，Percy Alexander（麦克马洪，珀西 · 阿历山大），87，92
Marckert，Jean-François（马利克特，吉安-弗朗索），16
Markov(=Markoff)，Andrei Andreevich(Марков, Андрей Андреевич)（马尔可夫，安德烈 · 安德里维奇，）
the elder，process（较老的，过程），49
Markowsky，George（马科夫斯基，乔治），83
Mary，Saint(῞Αγια Μαρία Θεοτόκος, Παναγία, Παρθένος)（玛利，圣），62-63
Matchings（匹配），4，6，33
Mate of a bit string（二进位串的配偶），41
Mate of an arc node（有向边节点的配偶），26
Matrix tree theorem（矩阵树形定理），43，44，100
Matsunaga，Yoshisuke（松永良弼），66
Maximal chains（极大链），36，38
Mayeda，Wataru（前田渡），73
McLean，Iain Sinclair（麦克里安，伊爱恩 · 辛克莱尔），58
Medicine（医疗，医药），54，57
Melodies（旋律），53，60，67
Mems(μ)：Memory accesses（内存访问次数），30-31，38，81，96，98
Mersenne，Marin（梅森，马林），53，67，74
Meters，poetic（诗的韵律），49-52，54，62-65，70，74
Metrical feet（韵律音步），51，63，74
Mikami，Yoshio（三上义夫），54
Mirror image（镜面映象），6，33，77，81，94
Mittag-Leffler，Magnus Gösta(=Gustaf)（米塔-列夫勒，马格努斯 · 戈斯塔），70
Mixed-radix number systems（混合进制数系），108
Modular Gray code for tuples（元组的模态格雷码），43
Mohar，Bojan（莫哈，波吉安），103
Moivre，Abraham de（莫里亚，德 · 阿布拉罕），13，68-70，75
Monomial symmetric functions（单调对称函数），68
Monotone Boolean functions（单调布尔函数），22，42
Montmort，Pierre Rémond de（蒙特莫特，皮埃尔 · 里蒙德的），68，75
Morphic sequence（形式系列），94
Morse，Samuel Finley Breese，code（莫尔斯，萨缪尔 · 芬利 · 布里斯码），50，70
Multicombinations:Combinations with repetition（多重组合：允许重复的组合），55-56，62，74
Multinomial coefficients（多项式系数），67
Multinomial theorem（多项式定理），68，70，75
Multipartitions:Partitions of a multiset（多重分划：多重集合的分划），67，75
Multiset combinations（多重集合的组合），74，111
Multiset permutations（多重集合的排列）53，62，69
Murasaki Shikibu(=Lady Murasaki)（紫式部），65
Music（音乐），50-53，59-60，67
notation（记号），52-53
rhythm（韵律），50-52，60
Myers，Eugene Wimberly，Jr（小迈尔斯，尤金 · 温贝利），99

N

n-cube（*n*立方体），30-31，45，101
Nakagawa，Noriyuki（中川敬之），73
Nārāyaṇa Paṇḍita, , son of Nṛsiṁha (नारायण पण्डित, नृसिंहस्य पुत्रः)（纳拉燕娜 · 潘迪塔，尼尔西姆哈的儿子），50，53，61-62，69，74
Natural correspondence（自然对应；参见左儿子/

右兄弟链接），3，34;*see*（参见）
Left-child/right-sibling links（左儿子/右兄弟连接）
Near trees（接近树），25-30，43，98
Near-perfect Gray code for nested parentheses（嵌套括弧的近乎完美格雷码），8
Needham，Joseph（李约瑟），49
Nested parentheses（嵌套括弧），2-8，12-15，17-18，21，33-34，39，40，71-73，76，82
Netto，Otto Erwin Johannes Eugen（尼特托，奥托·厄尔文·约·梅尼斯·尤金），70-72
Neuman，František（纽曼，弗朗梯斯克），46
Newton，Isaac，identities（牛顿，伊萨恒等式），101
Nievergelt，Jürg（尼维尔杰尔特，尤格），72
Nijenhuis，Albert（尼珍休斯，阿尔伯特），43，72
Noncommuting variables（非对称变量），110
Noncrossing chords（非交叉弦），6，33
Noncrossing partitions（非交叉分划），35，73，75
Nooten，Barend Adrian Anske Johannes van（奴廷，巴仁德·亚德里安·安斯克·约翰尼斯·凡），50
Notational conventions:（记号约定）
F^D, 34.
F^R, F^T, 33.
$F \lessdot F'$, $F \dashv F'$, $F \subseteq F'$, 35–37, 82.
$G + H$, $G \mp H$, $G \times H$, $G \diamond H$, $G \circledast H$, v.
Nucleotides（核苷酸），73
Null case（空的情况），61
Numbers，Catalan（卡塔兰数），2
Numerals，Arabic（数字，阿拉伯），55，74
Roman（罗马数字），109
Sanskrit（梵文数字），53
Nylan，Michael（尼兰，迈克尔），49

O

Odd permutations（奇排列），54，74
Offord，Albert Cyril（奥弗德，阿尔伯特·西利尔），96
Optimum algorithm（最优算法），23
Ord-Smith，Richard Albert James(=Jimmy)（奥德-史密斯，理查德·阿尔伯特·詹姆斯），70
Order ideals（顺序理想），82
Ordered forests（顺序森林），71
Ordered partitions(compositions)（顺序分划(组成)），50，54，74，111
Ordered trees（已排序的树），71，73
Organ-pipe order（手风琴顺序），98
Organic illustrations（有机的说明），47
Oriented forests（有向森林），23-24，42
Oriented spanning trees（有向生成树），43-44，102-103
Oriented tree numbers，table（有向树个数，表），23
Oriented trees（有向树），23-24，42-44，71，73
Out-degree of a vertex（顶点的出度），44，102
Outline of a forest（森林的外观），15

P

P_n (path graph)（P_n（通路图）），31，43，45，98，101，104
Pan-digital puzzles（泛数字难题），46-47
Panholzer，Alois（潘霍尔泽，阿罗伊斯），91，93
Parent pointers（双亲指针），23-24，32-33，42
Parentheses（括弧），2-8，12-15，17-18，21，33-34，39，40，71-73，76，82
Parking problem（停车问题），84，100
Partitions（分划），1-2，70
indecomposable（不可分解），75
noncrossing（非交叉），35，73，75
of an integer（整数），1，67-70，75
of a multiset（多重集合），67，75
of a set（集合），2，35，65-67，73，75
ordered，*see* Compositions（有序的分划，参见组成）
Party games（晚会游戏），56，65
Path graph(P_n)（通路图(P_n)），31，43，45，98，101，104
Path length（通路长度），90
Pendant vertex，*see* Endpoint（吊灯顶点，参见端点）

Pentagons（五边形），37
Perfect Gray code for nested parentheses（嵌套括弧的完美格雷码），8，39，82
Permutahedron（排列晶面体），37
Permutation representation of binary trees（二叉树的排列表示），38
Permutations（排列），1，10，52-54，62，70
  cyclic（循环），38
  descents of（下降），41
  even and odd（奇和偶），54，74
  flip（触发），84
  indecomposable（不可分解），106
  inverses of（逆），10
  null（空），61
  of a Latin verse（拉丁诗句），62-65，74
  of a multiset（多重集合），53，69
  restricted（限制），63-65，74
Pi(π)，as “random” example（作为“随机”例子的圆周率π），21，41，48，89，106
Piṅgala, Ācārya (आचार्य पिङ्गल)（平加拉，亚卡里亚），49-50
Plain changes（平易的改动），9，53，60
Plato=Aristocles，son of Ariston(Πλάτων = Ἀριστοκλῆς Ἀρίστωνος)（柏拉图），2
Poetry（诗），56，67
  meters for（节奏），49-52，54，62-65，70，74
  rhyme schemes（韵律方案），67，75
Poinsot，Louis（泊恩索特，卢伊斯），74
Polish prefix notation（波兰前缀记号），34，73
Polyhedra（多边形），37
Polyphase sorting（多阶段排序），50
Positive semidefinite matrices（正半定矩阵），100
Postorder（后根顺序），3，6，9，10，33，76，77，81
Postpreorder（后前根顺序），31-33，45
Power of a graph（图的幂），32，46
Prākṛta Paiṅgala (प्राकृत पैङ्गल)（普拉克尔塔·潘加拉），50，74
Preferential arrangements，*see* Weak orderings（优先安排，参见弱顺序）
Preorder（前根顺序），3，6，10-11，23，24，33-37，40，42，73，76，81，82，98
Preorder degree sequence（前根顺序的度序列），34
Prepostorder（前后根顺序），31-33，45-46
Prestet，Jean（普里斯特，吉安），64，75
Prins，Geert Caleb Ernst（普林斯，吉尔特·卡勒布·恩内斯特），71
Prodinger，Helmut（普罗丁格尔，赫尔穆特），91，93
Proskurowski，Andrzej（普罗斯库罗夫斯基，安德列），8，39
Prosody（普罗索迪），49-51，63-65，74
Proteins（蛋白质），73
Proteus verses（变幻无常的诗句），63，64，74
Prune-and-graft algorithm（剪枝和嫁接算法），11，35
Pseudo-complement in a lattice（格中的虚拟求补），84
Ptolemy，Claudius，of Alexandria (Πτολεμαῖος Κλαύδιος ὁ Ἀλεξανδρινός)（阿历山大的普托里米，克劳弟乌斯），63
Pun resisted（被抵制的双关语），32
Puteanus，Erycius(=de Putte，Eerrijk)（普蒂亚奴斯，埃里西乌斯(=德·普梯，埃里伊克)），62-64，74，109
Puttenham，George and/or Richard（普廷哈姆，乔治和/或理查德），66-67，75
Pyrrhics（皮尔里克斯），51-52，63，74

Q

q-ballot numbers（*q*选票数），38-39
q-Catalan numbers（*q*卡塔兰数），38-39
q-nomial coefficients（*q*项式系数），88
Quasi-Gray codes（拟格雷码），31-33

R

Rabbinic script（犹太法学博士的手稿），52
Radix-2 arithmetic（进制2算术），61
Radix-2 number system（进制2数系），49，52

PDG

Radix-3 number system（进制3数系），49

Ramesh，Hariharan (ரமேஷ் ஹரிஹரன்)（拉米斯，哈尔哈兰），99

Random binary tree（随机二叉树），16-19，40，47

Random forest（随机森林），15-16，40

Random oriented tree（随机有向树），43

Random Schröder tree（随机施罗德树），93

Raney，George Neal（拉尼，乔治·尼尔），40

Ranking（排阶），39，41，49-50，53，62，73，74

Rashed，Roshdi(=Rashid，Rushdi)(رشدي راشد)（拉西德，罗斯迪），55，107

Rational arithmetic（有理算术），105

Reachability test（可达性检测），97

Recurrence relations（递归关系），84-86，91，93，104，108-109

Recurrent states（递归状态），44

Recursion（递归），25-26

Recursion tree（递归树），13-14

Recursive algorithms（递归算法），73，75

Recursive procedure（递归过程），97

Recursive structure（递归结构），5，13，76，85

Reflected Gray code（反射格雷码），9

Reflection of a forest（森林的反射，参见共轭），33，*see* Conjugate

Regular graph（正则图），45

Regular polygon（正多边形），37

Reingold，Edward Martin (משה בן חיים ריינגולד, יצחק)（奈英高德，爱德华·马丁），72

Relative complement（相对补），22

Rémond de Montmort，Pierre（里蒙德·德·蒙特莫特，皮埃尔），68，75

Rémy，Jean-Luc（勒米，吉安·鲁克），18，40，92

Restricted growth sequences（限制增长序列），81，110

Resultants（结式），101

Reverse colex order（颠倒的协词典顺序），62，108

Revolving-door Gray codes（转动门格雷码），25，29，30，43

near-perfect（近乎完美），8

strong（强），99

Rhyme schemes（押韵方案），67，75

Rhythms（韵律），50-52，60

Richards，Dana Scott（理查德，达纳·斯戈特），34，38

Right path length（右通路长度），89

Right-child/left-sibling links（右儿子/左兄弟链接），7，34，38，78，81-82

Right-sibling /left-child links（右兄弟/左儿子链接），3，7，9，32，77，78，81-82，103

Riordan，John（里奥丹，约翰），88

River flows（河流），105

Robinson，Gilbert de Beauregard（鲁滨逊，吉尔伯特·德·比奥里加德），41

Rodrigues，Benjamin Olinde（罗德里格斯，本杰明·奥林德），18

Roelants van Baronaigien，Dominique（卢兰特斯·凡·巴罗奈吉安，多米尼格），8，9，86

Roman numerals（罗马数字），109

Rooted unlabeled trees，*see* Oriented trees（带根无标号树，参见有向树）

Roots of a forest（森林的根），3

Rossin，Dominique Gilles（罗森，多米尼格·吉尔斯），100

Rotation lattice，*see* Tamari lattice（转动格，参见塔马里格）

Rotations in a binary tree（二叉树中的转动），8-11，82

Rothe，Heinrich August（罗瑟，亨里茨·奥古斯特），70

Run-length coordinates d_k（运程长度坐标d_k），4，5，33，39，76，79

Ruskey，Frank（拉斯基·弗朗克），iv，8，9，14，16，39，73，86，97

S

S_n (Schröder number)（S_n（施罗德数）），41

Sachs，Horst（萨茨斯·霍斯特），101，103

Sack，Jörg-Rüdiger Wolfgang（萨克，约格–鲁迪吉尔·沃尔夫冈），92

Saka, Masanobu（坂正永）, 66

Sandpiles（沙堆）, 44, 103

Sanskrit（梵文）, 49-51, 53, 54

Śārṅgadeva, son of Sodhaladeva (शार्ङ्गदेव, सोढलदेव पुत्रः)（萨尔恩加迪瓦，索达拉迪瓦的儿子）, 53, 62, 108

Savage, Carla Diane（萨瓦吉，卡拉·戴安尼）, 72

Scaligero, Giulio(=Scaliger, Julius Caesar)（斯卡里吉罗，久利奥(=斯卡里吉尔，尤利乌斯，凯沙尔)）, 63

Schensted, Craige Eugene(=Ea Ea)（斯承斯德，克赖伊格·尤金(=雅·雅)）, 41

Schillinger, Joseph Moiseyevich (Шиллингер, Иосиф Моисеевич)（斯希令格，约瑟夫·芙伊西耶维奇）, 59-60, 74

Schooten, Frans van（斯库廷，佛兰斯·凡）, 60-61, 74

Schott, René Pierre（斯库特，里尼·皮埃尔）, 92

Scoins, Hubert Ian（斯科因斯，休伯特·伊安）, 73

Schröder, Ernst, numbers（施罗德·恩斯特数）, 41, 104

trees（树）, 41, 93

triangle（三角）, 93

Schwenk, Allen John（斯温克，阿兰·约翰）, 104

Scoins, Hubert Ian（斯科因斯，哈伯特·伊安）, 23

Scope coordinates（范围坐标）, 10, 36

SCOPE links（范围链接）, 6

Sedgewick, Robert（塞德杰维克，罗伯特）, 72

Sefer Yetzirah (ספר יצירה)（西佛·耶特吉拉）, 52

Sekanina, Milan（西卡尼纳，米兰）, 32

Seki, Takakazu（关孝和）, 54, 66, 74

Self-conjugate forests（自共轭森林）, 38, 78

Self-dual forests（自对偶森林）, 38

Self-transpose forests（自转置森林）, 38

Semba, Ichiro（仙波一郎）, 4

Semidistributive laws（半分配律）, 37

Semimodular law（半模律）, 36

Series -parallel graphs（串-并行图）, 27-30, 43, 93

Set partitions（集合分划）, 2, 35, 65-67, 73, 75

Seven deadly sins（七个致命的罪恶）, 57-58

Sha 'ari Tzedeq (שערי צדק)（沙阿利·钛泽德格）, 108

Shao Yung（邵雍）, 49

Shape of a random binary tree（随机二叉树的形状）, 16-17

Shape of a random forest（随机森林的形状）, 15-16

Shrinking an edge（边的收缩）, 25-27, 93

Sims, Charles Coffin, table（西姆斯，查尔斯·科芬表）, 107

Singh, Parmanand (परमानंद सिंह)（辛格，帕尔玛南德）50, 61

Skarbek, Władysław Kazimierz（斯卡贝克，伏拉弟斯塔夫·卡齐米尔兹）, 6

Sleep, Michael Ronan（斯立普，迈克尔·罗南）, 15

Smith, Malcolm James（史密斯，马尔柯姆·詹姆斯）, 25, 26, 29, 30

Socrates, son of Sophroniscus of Alopece (Σωκράτης Σωφρονίσκου Ἀλωπεκῆθεν)（苏格拉底，阿罗庇斯的索普罗尼斯库斯的儿子）, 2

Spanning arborescences, *see* Oriented spanning trees（生成树形图，参见有向生成树）, 43

Spanning trees（生成树）, 24-31, 43-45, 73

enumeration of（枚举）, 44-45

Spectrum of a graph（图的谱）, 101, 103

Sperner, Emanuel（斯珀纳，厄玛奴尔）, 21

theorem（定理）, 21, 41, 95

Spondees（斯篷迪斯）, 51-52, 63, 74

Sprugnoli, Renzo（斯普鲁格诺里，仁佐）, 39

Square of a graph（图的平方）, 32, 46

Squarefree integers（无平方整数）, 67

Stable states（稳定状态）, 44

Stanford GraphBase（斯坦福图库）, ii, iii, 30, 98

Stanford University（斯坦福大学），iv，19，72
Stanley，Richard Peter（斯坦利，理查德·彼得），1，2，38，61，82，93
  lattice（格），36-38
Star graphs（星形图），102
Stinson，Douglas Robert（斯廷森，道格拉斯·罗伯特），73
Stirling，James（斯特林，詹姆斯），66
  subset numbers（子集数），66-67
Strahler，Arthur Newell（斯特拉勒，阿瑟·尼维尔），47，105
Strong product of graphs（图的强积），v，45
Strong revolving-door order（强转动门顺序），99
Sum of graphs（图的和），v，45
Super-root of a forest（森林的超根），32，45
Superedge of a graph（图的超边），27-30，93
Suśruta (सुश्रुत)（苏斯鲁塔），54
Swetz，Frank Joseph（斯维特兹，弗朗克·约瑟夫），49
Symmetric order (inorder)（对称顺序(中根顺序)），3，9，10，17，38，39，77，82，83
Symmetric polynomials（对称多项式），110
Syntax，context-free（语法，上下文无关），46

T

t-ary trees（*t*叉树），40，85，90
  complete（完全），46
  random（随机），40
Tableaux（表景），38，41，85，95
Tacquet，André（塔尔奎特，安德尔），60
Tamari，Dov（塔玛利，多夫），82
  lattice（格），36-37，83，85
Tang，Changjie（唐常杰），8
Tartaglia，Niccolò Fontana（塔尔塔格里亚，尼科罗·冯塔纳），61
Tastes（风味），54
Tengbergen，Cornelia van Ebbenhorst（廷格贝尔根，科尔尼里亚·凡·埃伯本霍斯特），19
Ternary trees; *see also* *t*-ary trees（三叉树，也见*t*叉树），34，38，49
Thimonier，Loÿs（蒂莫尼尔，罗伊斯），107
Three-valued logic（三值逻辑），58
Threshold functions（阈值函数），23，95
Tompkins，Charles Brown（汤普金斯，查尔斯·布朗），72
Toppling（倒坍），44
Torus graphs（圆环体图），30，31，45，100，103
  directed（有向），103
Tot tibi...（诗句），62-65，74-75
Tournament digraphs（淘汰赛有向图），102
Transitive tournaments（传递淘汰赛），102
Transpose of a forest（森林的转置），33-34，38
Transpositions:Cyclic permutations of order（转置：顺序的循环排列）2，38
Traversal of a binary tree（二叉树的遍历），31-33
Tree representation of a series-parallel graph（串一并行图的树形表示），28，43
Trees（树），2-47，70-73
  binary，*see* Binary trees（二叉，参见二叉树）
  binomial（二项式），46，104
  Fibonacci（斐波那契），46，47
  free（自由），24，42-43，46，71，84，102，103
  Gray codes for（格雷码），8-11，35
  lattices of（格），35-38
  oriented（有向），23-24，42-44，71，73
  random（随机），14-19，40，43，47，93
  Schröder（施罗德），41，93
  spanning（生成），24-31，43-45
  t-ary（*t*叉），34，38，40，85，90
  traversal of（遍历），31-33
Triangular grids（三角形栅格），82
Triangularizing a matrix（三角化矩阵），99
Tribonacci sequence（三重斐波那契序列），50，108
Triply linked forest（三重链接森林），32-33，42，80
Trivial trees（平凡的树形），78
Trochees（扬抑格），51，63
Tuples（元组），1，48-52，61
Twelvefold Way（十二折法），1，61
Twisted binomial trees（扭曲的二项式树），46

Tyler, Douglas Blaine（蒂勒，道格拉斯·布莱尼），38

U

Uneasy nodes（非省事节点），29，43
Unit vectors（单位向量），42
Universal cycles（万有循环），60，74
Unlabeled free trees（未标号的自由树），24，42
Unlabeled rooted trees, *see* Oriented trees（未标号的带根树，参见有向树）
Unlucky nodes（非幸运节点），45
Unranking（未排阶），14，39，49-50，53，62，73，74
Unrooted trees, *see* Free trees（无根树，参见自由树）
Ushijima, Kazuo（牛岛和夫），8

V

Vacillating tableau loops（振动的表景循环），110
Vakhovsky, Evgenii Borisovich (Баховский, Евгений Борисович)（瓦科夫斯基，埃弗根里·皮里素维奇），101
van Baronaigien, *see* Roelants van Baronaigien（凡·巴罗奈尼恩，参见鲁兰特兹·凡·巴罗奈吉恩）
van Ebbenhorst Tengbergen, Cornelia（凡·埃布本霍斯特·廷格伯根·科尔尼里亚），19
van Nooten, Barend Adrian Anske Johannes（凡·奴廷，巴仁德·阿德里安·安斯克·约翰尼斯），50
van Schooten, Frans（凡·斯库廷，弗朗斯），60-61，74
Variations（方差），61
Vector spaces（向量空间），42
Vedic chants（吠陀的赞美诗），49
Venice, doge of（威尼斯，总督），57
Vergil(=Publius Vergilius Maro)（弗吉尔（=帕布里尤斯·弗吉里尤斯·马罗）），63
Vices（罪恶），57-58
Viennot, Gérard Michel François Xavier（维恩诺特，杰拉德·迈克尔·弗朗索斯·扎维尔），105
Virgin（美德），62-63
Virtues，55-58，62-63，74
Vo, Kiem-Phong（和金莲），95
von Christ, Wilhelm（冯·克里斯特，威廉），52
von Dyck, Walther Franz Anton（冯·带克，沃尔特·弗朗兹·安通），72
von Ettingshausen, Andreas（冯·爱丁绍森，安德列亚斯），70
Voting（投票），13，58，96

W

Wallis, John（沃利斯·约翰），53，61，64，67，75
Warren, Jon（沃仁，琼），16
Watanabe, Hitoshi（渡部和），73
Watson, George Neville（沃特森，乔治·尼维里），88
Weak orderings（弱顺序），71
Wells, Mark Brimhall（威尔斯，马克·布里姆哈尔），72，73
Wen, King of Chou（姬昌=周文王），48-49，74
Wheel graph（轮图），44
Wheels, concentric（轮，同心），59
Whitworth, William Allen（维特沃思，威廉·阿伦），64-65，75
Wibold, bishop of Cambrai(=Wiboldus, Cameracensis episcopus)（维博尔德，康布莱的主教(=维博尔度斯，卡米拉森西斯·埃皮克索普斯)），55-57，67
Wilf, Herbert Saul（维尔夫，希尔伯特·绍尔），43，72
Winkler, Peter（温克尔，彼得），93
Wolf, Margarete Caroline（沃弗，玛加里特·卡罗里尼），110
Worm's walk（蚯蚓步），3，14-16，33，76，82，91，94

X

Xiang, Limin（相利民），8

Y

Yang Hsiung（杨雄），49-50
Yano，Tamaki（矢野环），65，66
Yijing，*see I Ching*（易经）
Yin and yang（阴阳），48-49
Young，Alfred，tableaux（扬，阿尔弗雷德表景），38，41，85，95

Z

Zaks，Shmuel (שמואל זקס)（扎克斯·斯缪尔）33，34，38，79，84
Zhou Wenwang，*see* King Wen of Chou（周文王）
Ziegler，Günter Matthias（齐格勒，库特·玛蒂亚斯），83